Bio-inspired Dynamic Control Systems with Time Delays

DERRICK, BENJAMIN,JOHN

Bio-inspired Dynamic Control Systems with Time Delays

Benjamin John Derrick

A Thesis submitted for the degree of
Doctor of Philosophy

School of Engineering and Computing Sciences
University of Durham
England

April 2014

Bio-inspired Dynamic Control Systems with Time Delays

Benjamin John Derrick

Submitted for the degree of Doctor of Philosophy

2014

Abstract

The world around us exhibits a rich and ever changing environment of startling, bewildering and fascinating complexity. Almost everything is never as simple as it seems, but through the chaos we may catch fleeting glimpses of the mechanisms within. Throughout the history of human endeavour we have mimicked nature to harness it for our own ends. Our attempts to develop truly autonomous and intelligent machines have however struggled with the limitations of our human ability. This has encouraged some to shirk this responsibility and instead model biological processes and systems to do it for us.

This Thesis explores the introduction of continuous time delays into biologically inspired dynamic control systems. We seek to exploit rich temporal dynamics found in physical and biological systems for modelling complex or adaptive behaviour through the artificial evolution of networks to control robots. Throughout, arguments have been presented for the modelling of delays not only to better represent key facets of physical and biological systems, but to increase the computational potential of such systems for the synthesis of control.

The thorough investigation of the dynamics of small delayed networks with a wide range of time delays has been undertaken, with a detailed mathematical description of the fixed points of the system and possible oscillatory

modes developed to fully describe the behaviour of a single node. Exploration of the behaviour for even small delayed networks illustrates the range of complex behaviour possible and guides the development of interesting solutions.

To further exploit the potential of the rich dynamics in such systems, a novel approach to the 3D simulation of locomotory robots has been developed focussing on minimising the computational cost. To verify this simulation tool a simple quadruped robot was developed and the motion of the robot when undergoing a manually designed gait evaluated. The results displayed a high degree of agreement between the simulation and laser tracker data, verifying the accuracy of the model developed.

A new model of a dynamic system which includes continuous time delays has been introduced, and its utility demonstrated in the evolution of networks for the solution of simple learning behaviours. A range of methods has been developed for determining the time delays, including the novel concept of representing the time delays as related to the distance between nodes in a spatial representation of the network. The application of these tools to a range of examples has been explored, from Gene Regulatory Networks (GRNs) to robot control and neural networks. The performance of these systems has been compared and contrasted with the efficacy of evolutionary runs for the same task over the whole range of network and delay types.

It has been shown that delayed dynamic neural systems are at least as capable as traditional Continuous Time Recurrent Neural Networks (CTRNNs) and show significant performance improvements in the control of robot gaits. Experiments in adaptive behaviour, where there is not such a direct link between the enhanced system dynamics and performance, showed no such discernible improvement. Whilst we hypothesise that the ability of such delayed networks to generate switched pattern generating nodes may be useful in Evolutionary Robotics (ER) this was not borne out here.

The spatial representation of delays was shown to be more efficient for larger networks, however these techniques restricted the search to lower complexity solutions or led to a significant falloff as the network structure becomes more complex. This would suggest that for anything other than a simple genotype, the direct method for encoding delays is likely most appropriate. With proven benefits for robot locomotion and the open potential for adaptive behaviour delayed dynamic systems for evolved control remain an interesting and promising field in complex systems research.

Declaration

The work in this thesis is based on research carried out in the School of Engineering and Computing Sciences at the University of Durham. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Acknowledgements

I would like to acknowledge my late principal supervisor Professor Valentin Vitanov for his help, inspiration and constant support without which I would have never started on this journey. He was a true gentleman and is sorely missed. Many thanks are due to Professor Alan Purvis who kindly consented to take up the reins after Val passed away under difficult circumstances. His advice and encouragement were vital in the completion of this work.

In addition, there are several other people to whom I owe a debt of gratitude for their help be it of a technical, grammatical or emotional nature. Thanks go to Dr Richard McWilliam, Mr Joshua Cowling, my family for their love and support and my wonderful wife Alexandra, for without her none of this would have been possible.

Contents

List of Tables

List of Figures

Nomenclature

α	gradient variable in $S1$, page 213
α	heading angle, page 145
α	significance level, page 205
α,β,γ	single leg driving angles, page 105
α_l, β_l,γ_l	joint angles for leg l, page 235
α_r	sensor absolute angle, page 147
α_s	sensor relative body angle, page 147
$\bar{R}_j$	average ranking of group j, page 208
$\bar{R}_j$	average ranking of group j, page 214
$\bar{R}_j$	average ranking of group j, page 237
$\bar{x}(I, w, \theta)$	approximation for fixed point in stability region, page 93
β	gradient variable in $S2$, page 213
β	recurrent connection weight, page 53
$\boldsymbol{\gamma}$	right hand side of the acceleration equation, page 107
$\boldsymbol{\tau}$	diagonal time constant matrix, page 44
$\boldsymbol{\theta}$	threshold vector, page 44
χ	3D rotation, page 263
χ^2	chi-squared distribution, page 205
Δt	time step, page 100
ϵ	coefficient of restitution, page 114
η_c	maximum distance between objects in static contact, page 113

Γ	density, page 121
Γ	fundamental synaptic time delay, page 188
$\hat{\mathbf{n}}$	surface normal vector, page 112
κ	neuron time constant, page 53
λ	anti-'popping' factor, page 113
$\mathbf{A}$	rotational transformation matrix, page 118
$\mathbf{a}$	example vector of length 3, page 260
$\mathbf{a}_0$	contact attachment point vector, page 117
$\mathbf{b}$	example vector, page 260
$\mathbf{b}$	wall end point, page 147
$\mathbf{c}$	wall end point, page 147
$\mathbf{d}$	sensor position vector, page 147
$\mathbf{e}$	sensor ray/wall intersection, page 147
$\mathbf{F}^{\mathbf{A}}$	applied force vector, page 107
$\mathbf{F}_n$	normal force vector, page 116
$\mathbf{F}_s$	virtual spring force vector, page 117
$\mathbf{F}_{fr}$	friction force vector, page 116
$\mathbf{G}$	3×4 matrix that depends on the Euler parameters, page 109
$\mathbf{I}^{\mathbf{ext}}$	external input vector, page 44
$\mathbf{I}^{\mathbf{int}}$	total internal input vector, page 44
$\mathbf{J}'$	inertia matrix in the body frame, page 107
$\mathbf{k}$	unit vector in z-direction of ground plane, page 115
$\mathbf{L}$	angular momentum vector, page 114
$\mathbf{M}$	mass matrix, page 107
$\mathbf{n}'$	applied torque vector in the body frame, page 107
$\mathbf{p}$	2D position vector of robot, page 147
$\mathbf{p}$	Euler parameter vector, page 109
$\mathbf{p}$	current point position vector, page 117

$\mathbf{p}$	linear momentum vector, page 114
$\mathbf{r}$	vector to centroid of body, page 107
$\mathbf{r}^P$	position vector of point P in the ground fixed reference frame, page 262
$\mathbf{s}'^P$	position vector of point P in the body reference frame, page 262
$\mathbf{s}'_f$	body frame foot position vector, page 111
$\mathbf{W}$	connection weight matrix, page 44
$\mathbf{y}$	output vector, page 44
$\mathcal{O}$	order of, page 99
$\mathcal{R}$	set of real numbers, page 100
$\mathcal{Z}$	set of natural numbers, page 100
$\mathrm{lb}(w, \theta)$	left boundary of cusp, page 87
$\mathrm{rb}(w, \theta)$	right boundary of cusp, page 87
μ	coefficient of friction, page 116
μ_k	coefficient of kinetic friction, page 116
μ_s	coefficient of static friction, page 116
ω	rotational speed, page 109
ω'	body rotational speed vector, page 107
Φ_r	constraint matrix, page 107
$\Phi_{\pi'}$	constraint matrix, page 107
$\sigma(x)$	standard sigmoid function $\sigma(x) = 1/(1 + e^{-x})$, page 46
τ	self-recurrent connection time delay, page 77
τ_i	time delay, page 73
τ_s	synapse time delay, page 53
τ_{ij}	continuous delay between neurons i and j, page 74
τ_{ij}	delay parameter, page 46
θ	node bias, page 77
θ_j	population media for the jth group, page 211

a_n stable value reached for oscillation $n \in \mathbb{Z}$, page 93

a_{ij} weight of connection between neuron i and j, page 53

c abbreviation of trigonometric cosine function, page 111

c number of connections, page 181

c_f scent contribution from food f, page 213

D scent diffusion radius, page 213

d distance of interpenetration, page 112

d_f distance of food f from the sensor, page 213

df statistical degrees of freedom, page 205

dt simulation time step, page 112

dx movement in x per timestep, page 126

F food items remaining in the environment, page 213

F force applied to body, page 109

F vector valued smooth continuous function, page 73

f fitness value, page 161

f intermediate variable in sensor calculation, page 147

$f(t)$ smooth differentiable function, page 100

F_n normal force, page 116

f_p penalty force, page 112

F_{fr} friction force, page 116

f_{max} maximum fitness, page 224

g acceleration due to gravity, page 112

h initial height, page 223

h sensor intersection fraction, page 147

h step size, page 99

H_0 null hypothesis, page 208

H_1 alternative hypothesis, page 208

I impulse, page 115

$I(t)$ current into neuron, page 44

$I(t)$ node external input, page 77

I_f impulse for foot f, page 115

$I_i(t)$ external dynamical input function to neuron i, page 46

$I_i(t)$ external input to neuron i, page 74

J' element of inertia matrix in body frame, page 109

k number of independent samples, page 205

k_d damping coefficient, page 112

k_f virtual spring coefficient, page 117

k_i integral constant, page 113

k_s spring coefficient, page 112

KW result of Krustal-Wallis test, page 205

L angular momentum, page 114

l left motor speed, page 145

l limb length, page 126

l_i length of limb i, page 105

l_{ij} length of connection between node i and node j, page 181

m mass, page 108

m multiplier value, page 161

m_l left motor speed, page 161

m_r right motor speed, page 161

N number of neurons in the system, page 46

N number of nodes in the system, page 50

N total number of data, page 208

N total number of data, page 214

N total number of data, page 237

n number of steps in a gait, page 126

n' torque in body frame, page 109

n_c	number of contact points, page 115
n_c	number of food items consumed, page 214
N_r	number of regulatory (output) nodes in the system, page 50
N_s	number of structural (hidden) nodes in the system, page 50
o_1	output from motor neuron 1, page 161
o_2	output from motor neuron 2, page 161
o_i	output space response of motor neurons, page 74
$o_{l,1}$,$o_{l,2}$	the two motor neurons for leg l, page 235
p	penalty value, page 161
r	intermediate variable in the manual design of a gait, page 126
r	position of centroid of body, page 109
r	reward value, page 161
r	right motor speed, page 145
R_j	rank of group j, page 208
R_j	rank of group j, page 214
R_j	rank of group j, page 237
R_T	interpolation error, page 100
s	abbreviation of trigonometric sine function, page 111
s	sensor reading, either S_L or S_R, page 213
$S(v)$	sigmoidal mapping, page 44
$S1$	preprocessed sensor input, related to scent concentration, page 213
$S2$	preprocessed sensor input, related to distance from food source, page 213
s_d	distance of sensor from offset origin, page 147
S_L	left food scent sensor input, page 213
s_o	sensor longitudinal offset, page 147
S_R	right food scent sensor input, page 213
s_r	sensor distance, page 147

T	maximum duration of simulation, page 214
T	maximum run length, page 162
T	node time constant, page 77
t	current duration of run, page 162
t	simulation time, page 214
t_c	collision time, page 113
T_i	neuron time constant, page 46
T_i	time constant of each neuron, page 74
v	sensor direction indicator, page 147
$V(t)$	membrane potential, page 44
w	weight of self-recurrent connection, page 77
w_r	half inter-wheel distance, page 145
w_{ij}	weight matrix element connection neurons i and j, page 46
x,y,z	orthogonal Cartesian coordinate system, page 105
$x(t)$	response of node at time t, page 77
x^*	fixed point of x, page 78
x_i	total input to neuron i, page 46
$x_j(t)$	output of neuron x, page 53
$y_i(t)$	output of neuron i, page 46
$y_i(t)$	output of neuron i, page 74
z	height of body off ground, page 126
z	z score, used to denote a variable transformed into standard form, i.e. with mean zero and standard deviation one, page 211

Part I

Introduction

Introduction

We live in a complex and ever changing environment. Despite decades of intensive research, and promising initial progress, robots have only just begun to influence our every day lives. These devices carry out some of the simplest of our chores, with reasonable efficacy given a suitable structured environment, but do not live up to the enticing visions of intelligent and erudite robotic servants carrying out our every whim greatly anticipated across science fiction and popular culture. Why are these currently out of reach, and what can be done to develop useful robot controllers which are robust and flexible enough to function in our dynamic surroundings?

In the search for the machines, automatons and intelligent systems of the future, there is compelling evidence that human design may not be the best way. As systems have become more complex, the manual effort required to control them becomes drastically increased or indeed impossible, as decades of research have yet to yield technologies that allow robots to operate in the majority of human environments whilst being useful and attaining worth. Aside from science fiction, robots have only just started to influence our lives as toys and low level domestic aids, but we have not yet built anything approaching our aspirations for this technology.

> This limitation is not because of any deep mystery, but simply due to to the limitations of us poor humans in understanding complex systems [...] an evolutionary approach allows emulation without comprehension [46, p.3]

There is a tendency for researchers in Artificial Intelligence (AI) to see human behaviour as the desired outcome from their efforts. Even Brooks et al, who argued for a regression in design aim to focus on insect level creatures and to build up from there, quickly targeted the replication of human behaviour. This all, in some way, assumes that simulated human behaviour

is the goal of artificial intelligence, whereas the focus of this review is in the technology that will enable us to develop useful intelligent robots for use in the real world. Whether they have human-like behaviour or not is really neither here-nor-there. From the beginning of the move away from Good Old Fashioned Artificial Intelligence (GOFAI), Subsumption Architectures have demonstrated that systems do not have to be highly sophisticated to demonstrate intelligent behaviour, and that:

> Intelligence is in the eye of the observer [22, p.16]

We need to find the "juice of life" as proposed by Brooks [25, p.10], when he suggested that we may be missing something in the development of artificial creatures, as their biological counterparts are much more robust and learn much more quickly. Exciting developments in ER have produced a suite of richly dynamic evolved controllers, but the question remains as to whether these techniques are a way forward towards the intelligent, adaptive, autonomous systems of the future.

It has been argued that for the evolution of adaptive behaviour we must simply be able to harness and exploit the behaviour of dynamic systems across an appropriate range of time scales [45]. In the formulation of the CTRNN model so popular in ER, the dynamics of information propagation along synaptic connections is omitted. Whereas, in the related study of GRNs the contribution to the dynamics of such delays are widely accepted as important and are often included in the reconstruction and analysis of biological systems. By altering the governing equations of these systems, we transform them into Delay Differential Equations (DDEs) from Ordinary Differential Equations (ODEs) and enable the possibility of chaotic and oscillatory behaviour even for a single node. Through the introduction of even a single delay, due to the need to specify a continuous distribution of initial conditions, any network may be considered as an infinite dimensional system. This not only complicates any analysis or application of such systems

but also increases the richness of the dynamics that they may express and may be harnessed for our purposes.

Through the introduction of increased information processing and storage in the modification of synaptic connections, we hypothesise that simpler networks may be capable of behaviour previously impossible. In addition to this, the ability for a single node or small network to behave as pattern generators may facilitate the development of gaits for legged robots and interesting solutions for adaptive behaviour.

Research Questions

How does adding connection time delays into CTRNNs affect the dynamics of the system, and how can these additional parameters be governed by an Evolutionary Algorithm (EA)? With a focus on an application to the control of robotic systems, can this modified architecture be harnessed to measurably increase the ability of an Artificial Neural Network (ANN) to perform common tasks in ER (specifically adaptive behaviour and legged robot locomotion)? The potential fitness of a particular solution is arbitrarily defined based on the design of the experiment and so reference should be made to classical studies in ER and comparison drawn from unmodified solutions.

The Allure of Complex Systems

If I may be permitted, I would take a moment to address the motivation for the study of complex systems. They are difficult, and perhaps unsurprisingly complex, to comprehend, use, harness of otherwise investigate. They are non-linear, stochastic and indeterministic and are typically very difficult to analyse and synthesise. Despite this such systems are all around us and affect almost everything which we do and experience.

They require a whole new multi-disciplinary tool-kit from biology, mathe-

matics, computer-science, engineering, non-linear dynamics and chaos; merging together in an attempt to explore some of the most complex and fundamental aspects of life, the universe and everything. Nothing is simple or as it seems, but from that inherits a richness and elegance which is fascinating.

Contributions and Thesis Structure

This Thesis is structured as to divide it into a number of Parts. The nature of this infrastructure project supports a 'methods and materials' approach where the development work is presented grouped together and is then brought together to test and analyse the results; Part II and Part III respectively. Against the outline structure of the Thesis presented below the outline, role and key contributions of each element is stated.

Part I Introduction

The virtue of including time delays into dynamic networks for the purposes of developing adaptive behaviour has not hitherto been considered. Whilst delay systems are common in problems of classical control theory and biology, they have not yet been introduced into ER, nor have their dynamics considered as contributing usefully to the system beyond the simple preconditioning of sensor data within feed-forward discrete ANNs [34, 16].

Through the more detailed modelling of synaptic links it is hypothesised that the information processing capability for a network of a given size is increased, suggesting the possibility of evolved solutions with reduced complexity over that previously possible. Beyond this, this simple modification to the standard model permits the sustained evolution of pattern generation by even a single self-recurrent node. This alone would warrant investigation and application of such systems to the evolution of gaits and complex behaviour.

Part II Methods and Materials

This Part presents the development of several pieces of complementary research which are collectively required in order to support the investigation of research questions defined in Part I.

Chapter II.1 Background

This Thesis covers a wide range of interrelated topics which are introduced in this Chapter, highlighting the opportunities for novel investigation and preparing the background to the rest of the Thesis. Topics covered include adaption in natural and artificial systems, Artificial Neural Networks, Gene Regulatory Networks, dynamical analysis of complex systems and the evolution of adaptive behaviour.

Chapter II.2 Dynamics of Small Delayed Dynamic Networks

The inclusion of time delays within a continuous recurrent neural network architecture enhances an already rich set of dynamics and improves the capabilities of the system.

In this research a thorough evaluation and determination of the additional dynamics developed by small delay systems is presented for a modified Hopfield type recurrent dynamic neural network. Previous works have considered solely the non-delayed form of CTRNN, or the stability and mathematical characterisation for systems of DDEs modelling neural or gene regulatory networks, rather than the extra dynamic capabilities of interest in the context of ER.

This Chapter explores this difference in an intuitive, mathematical and numerical manner, comprehensively evaluating the dynamical differences made by the inclusion of delays, and highlights potential applications. Specifically, it is shown how even a single delayed node can oscillate in a governable manner and that complex switched oscillatory behaviour could be easily evolved. This evidence bolsters the case for investigating applications to robot locomotion in Chapter III.3 and justifies the development of the simulation tools and demonstration quadruped robot undertaken in Chapters II.3 and II.4 respectively.

Chapter II.3 Minimalistic Simulation of a Quadruped Robot

The research questions mandate study of application of dynamic neural networks with time delays to robot locomotion. Chapter II.1 introduces a variety of related literature and presents the advantages and disadvantages of a simulation based approach for the evaluation of a Genetic Algorithm (GA). Principal amongst the disadvantages is the computational effort required, and early

on in the field of ER, Jakobi [60] developed minimal simulations which captured the 'base set' of the environment which affected the behaviour of evolved individuals. Sympathetic with this view, this Chapter develops a computationally efficient Three-Dimensional (3D) simulation environment for evaluating the gait of quadruped robots. Results of trials were compared with laser tracker recordings of the demonstration quadruped, developed in Chapter II.4, with the simulation replicating the gait with a high degree of accuracy both with respect to the distance travelled and key characteristics of the motion. The validated simulation is used in the experimentation of Chapter III.3.

Chapter II.4 Model Verification using Real Robots

In ER evolved solutions are embodied in real or virtual systems, and it is in the interaction of morphology, control and environment that behaviour is emergent. Whilst the computational rigours of EAs often renders evolution in virtual environments more feasible than in hardware, it is necessary to capture the essence of the real system for successful evolution. This Chapter describes the modelling of real wheeled robots used in Chapters II.5 and III.2 and the development of a quadruped robot for the purpose of validating the simulation described in Chapter II.3.

Chapter II.5 Evolving Adaptive Behaviour with Time Delays

This Chapter demonstrates the successful evolution of simple adaptive behaviour in a well studied biologically inspired task for a two wheeled Khepera robot, using the model detailed in Section II.4.2. Evolved solutions are analysed for their sensitivity and reliance on time delays for achieving high fitness. For the purposes of comparison a traditional neural network model without delays was evolved for the same task and its performance analysed as a

range of delays were added to the system. Delays are shown to be highly significant for the maintenance of high fitness evolved behaviour and those systems evolved with delays are proven to be much more robust to changes which may render them more suitable for application in extreme environments. This T-junction task is then reused in Chapter III.2 for the investigation of the encoding methods developed in Chapter II.6.

Chapter II.6 Methods for Determining Time Delays

To evolve systems with delays they must be encoded within the genome of each individual in the simulation population. Beyond the simple direct encoding of these values, this Chapter introduces the concept of spatial representation of the network and the geometrical determination of the delays in the system. A range of methods are developed and analysed for the efficiency of the encoding and restrictions placed on the possible set of values delays may take.

A number of novel approaches for the encoding of delays within evolutionary algorithms have been proposed, with the core concept of spatial representation of network structures and geometrical determination of connection delays. The relative advantages and disadvantages are presented along with the relative suitability of each for a range of scenarios.

Part III Results and Analysis

In this Part the contributions of Part II are brought together and tested in an attempt to answer the research questions of Part I.

Chapter III.1 Introduction

To support the resolution of the hypotheses presented throughout and answer the research questions above, this Part presents the

results of evolving systems with and without delays determined by each method presented in Chapter II.6. The results of multiple runs are statistically processed to develop an understanding of the underlying distribution of achievable fitness for each method for a range of common examples in the field, split between adaptive behaviour and robot locomotion. Beyond simply considering fitness, the data are investigated for the impact of methods on the optimal structure of solutions.

To evaluate the merits or otherwise of the ideas presented in this Thesis, each of the delay determination techniques were applied to a wide ranging set of commonly studied tasks within the ER domain and statistically appropriate conclusions drawn from the results of repeated runs.

Chapter III.2 Adaptive Behaviour

Much of the work in ER has focussed on wheeled robots as animats to explore the emergence of adaptive behaviour (see Chapter II.1). This Chapter extends the T-junction test of Chapter II.5 to compare the various encoding methods of Chapter II.6 and also includes a example of chemotaxis.

Chapter III.3 Robot Locomotion

Another significant area of endeavour in this field is that of legged locomotion, as evidenced by the literature survey in Chapter II.1. ER studies have previously included both adaptive behaviour and legged locomotion in the same body of research, and thus the simulation environment and quadruped model developed in Chapters II.3 and II.4 are applied to the task of developing an effective quadruped gait. Initially however, the same leg model is applied to a single leg (as per Section II.3.1) as a fundamental building block.

Chapter III.4 Conclusions

This Chapter summarises the conclusions of the experiments carried out earlier in this Part, in Chapter III.2 and Chapter III.3. Through the novel introduction of connection time delays into Recurrent Neural Networks (RNNs) a small network was shown to be able to efficiently govern a single leg walking analogue which was previously impossible at this scale without an external input to the network. However, experiments in adaptive behaviour were unable to demonstrate any measurable increase in capability following the introduction of delays. Whilst all of the delay encoding methods proposed in Chapter II.6 were capable of evolving solutions for all of the experiments reported, there was no demonstrable benefit to the additional complexity and there was evidence that the evolutionary search was artificially constrained to lower complexity solutions for spatial representation methods.

Part IV Summary & Conclusions

This Part presents the conclusions drawn from the results and analysis developed throughout the Thesis, supporting or contradicting the hypotheses presented. Avenues for future exploration and the contributions of this research are also considered.

Part V Appendices

Part II

Methods and Materials

Chapter II.1

Background

Inspired by nature, we seek to evolve systems that develop emergent intelligence and adaptive behaviour. This requires a control architecture with the potential to develop this kind of behaviour and which is capable of being evolved artificially. Lipson [74] draws careful attention to the subtle distinction between *optimisation* and *synthesis*. Synthesis is inherently open ended and there is little or no knowledge of what components are required to reach an optimal solution, whereas optimisation revolves around the careful tuning of parameters to achieve optimality in a known system. The search space may be vast, but the process is intrinsically limited by an exhaustive search.

This distinction can clearly divide the field of ER, as much of the earlier work on the subject involved the selection of parameters for fixed ANNs, which is optimisation. More recent work has tended to focus on the more open-ended automatic definition of ANNs, which is synthesis, and it is most likely that through synthesis of complex systems useful results may be achieved. In work inspired by Sims [100, 99] interesting locomotor robot morphologies were successfully evolved using an L-system representation, specifying joints and their independent oscillatory properties. Initially, very simple evolutionary methods and architectures were evolved for very spe-

cific tasks. Evolutionary search is just that, a search process, which will ruthlessly exploit any and all factors within the system to achieve a high 'fitness' as described, for the most part, by simple mathematical functions. Without care, this can lead to evolved systems highly reliant on the specific circumstances of their evolution, and so it becomes necessary to instil generality into these systems. This generality may be attempted through careful consideration of the evolutionary process, and through learning and plasticity in the evolved controllers so that they have:

> ...the ability to carry out a certain task in different environmental conditions or the ability to carry out different tasks. .... Such systems will probably require more internal complexity that the simple non general systems which we described. [83, p.215]

The evolutionary methods used vary greatly from study to study. The simplest form is the one described above, where each individual of a generation is evaluated alone with respect to some 'fitness function', however there are many other methods designed to improve on or avoid certain pitfalls in specific circumstances. Not least among these is the definition of the fitness function, for without this there is no guidance for the evolutionary process. Traditionally these functions have been hand crafted by individual researchers based upon experience and not a little trial and error to achieve good results. For simple problems this is generally not a problem, but for more complex tasks the 'bootstrapping problem' is encountered. This is where the complexity of the task is such that any randomly generated initial population of individuals will attain uniformly negligible fitness and thus the evolutionary process will tend to random genetic drift. A common approach to the solution of this issue is in the incremental increase in the complexity of the evolutionary run from an initially simple problem for which an evolved solution may be found, tending towards the final form

required. This *incremental evolution* may be achieved through altering variously the fitness function, the environment, the robot morphology or control structure. However, the requisite manual decomposition in the majority of early work in this area imposes *a priori* constraints upon the final solution and may lead to overly complex or non-optimal solutions. For further details please see later sections of this Chapter, or Nolfi and Floreano [83] for a comprehensive introduction.

Much of the work thus far takes the form of a proof of concept, rather than the development of useful systems that could not be arrived at though other means. Almost two decades ago, Brooks stated that, "To compete with hand coding techniques it will be necessary to automatically evolve programs that are one to two orders of magnitude more complex than those previously reported in any domain"[24, p.3]. For the largest part, this holds true today, but significant progress has been made, and ER remains an exciting area of research.

This Thesis covers and develops a wide range of different, but related, areas of research. To establish the background to the work presented later, we must review a large body of literature which is separated into discrete Sections. Whilst a great deal of this material is generally applicable across the entirety of the Thesis there are some cases of direct read-across; Section II.1.3 applies almost entirely to Chapter II.2, Section II.1.4 largely informs Chapter II.5 and Section III.2, whereas Section II.1.5 applies to Chapter II.3 and Section III.3.

II.1.1 Adaption in Natural and Artificial Systems

The field of ER can be said to be based on the work of Holland [51], where he discussed the development of artificial adaptive systems. Initially highly specialised, the field has now broadened significantly with a wide body of research generated over the last two decades. ER aims to generate artificially

adaptive systems through the application of biologically inspired methods of evolution, breeding and selection; mimicking both the methods used and system structure from nature. Initially, search heuristics such as GAs were used to evolve computer programs, using LISP like languages. More recently, ER work has focussed on neural networks, particularly dynamically rich CTRNNs, as highly flexible noise tolerant control architectures. This kind of approach is necessary for its ability to generate solutions independent of the human architect of the evolutionary process, and has great potential for the control of highly complex or intelligent systems that may be very difficult, highly costly or simply impossible to design manually.

> The more we can understand of a system from the mechanical perspective, the less likely we are to attribute agency, personhood [sic], conciousness to it - which is why the ER approach that can produce comprehensible behaviour from incomprehensible mechanisms offers possibilities that the conventional design lacks. [46, p.10]

ER looks for emergent behaviour from the interaction of evolved control architectures with the environment, and is both a tool for the generation of control systems and for the study of cognitive processes, as artificial systems may be minutely dissected and analysed to build understanding of cognition and intelligence at the most fundamental level. The process starts with an initial population of 'individuals', representing particular instances of proposed architectures, which are then evaluated and the 'fittest' individuals pass on their genetic material. Over a phylogenetic time scale the process aims to find a near optimum solution. The concept is that the solution arrived at is entirely independent of the human designer, although *a priori* knowledge of a solution is almost inevitably embedded through the design and evaluation of the fitness function and parameters governing the evolutionary process.

Unfortunately the Law of Unintended Consequences holds significant sway in this field, and makes the task of the researcher that much more difficult because of it. In trying to govern and steer a complex evolutionary search with a formulaic fitness function and typically an approximation of embodiment for the system, the algorithm's exploitation of the search space will often lead to unexpected consequences. The design of the fitness function must therefore demand considerable thought and reflection in its design and implementation.

These methods are powerfully adaptive, and systems may develop the ability to learn or maintain internal states, but thus far have been limited to relatively simple problems (compared to those which naturally evolved biological systems, such as ourselves, encounter on a daily basis). In the anticipated resolution of these issues, ER is attractive because "... artificial evolution can develop mechanisms that go beyond reactive behaviour when this is necessary without being explicitly told to do so" [83, p.151].

Competitive co-evolution has demonstrated the ability to evolve controllers for more complex tasks due to the automatic task incrementation (without input from the designer of the experiment) as co-evolved individuals interact [83]. These methods however can suffer from the *Red Queen Effect*, named after a Lewis Carrol's fictional character who was always running forwards, but never made any progress as the landscape moved with her. In this context it refers to the progress of one species being negated by improvements in the other, leading to potential evolutionary stagnation and random genetic drift. Despite this, co-evolutionary methods have been shown to produce interesting results, and has significant benefits as a form of *intrinsically incremental evolution* [37]. However, this kind of approach is limited by the potential to structure the required problems in this form.

Whilst early work in GAs used simple direct encodings (See [83] for examples in ER), as techniques became more sophisticated real valued parameters

and variable length genotypes were introduced. Specialised algorithms were developed such as Genetic Programming (GP) [65], Cellular Encoding (CE) [42, 43] and Species Adaption Genetic Algorithm (SAGA) [45].

More recently, Stanley developed an algorithm called NeuroEvolution of Augmenting Topologies (NEAT) [107, 108, 104]. This open-ended evolutionary architecture evolves neural networks, beginning with a minimal genotype of solely input and output nodes arranged in a perceptron-like feed-forward network defined by the experimenter, where the complexity of the network topology may increase by inserting new neurons into existing connections or adding new connections in the network. It has been shown that for simple control tasks the NEAT algorithm is often more efficacious than other methods [107, 112].

An extension to this, called Hyper-NEAT [106], was developed for the evolution of very large scale neural networks using Compositional Pattern Producing Network (CPPN) [105]. These networks were proposed as a novel abstraction of a developmental encoding, mapping the phenotype without local interactions. They are essentially standard ANNs, where the activation function for each node can be non-linear (e.g. sigmoid), trigonometric or gaussian. By developing networks with, for example, two inputs, the evaluation of the CPPN over a continuous range generates a deterministic pattern which can be sampled at any resolution. Initially, they were used in genetic art where the pattern network is evolved to produce repeated and structural motifs often attributed to conventional developmental abstractions.[1] By representing the connections in a neural network as a four dimensional hyper-cube (four inputs of x_1,x_2,y_1,y_2), the weight of each connection was encoded in the pattern. This approach was applied to a number of examples and has the advantage in that the developmental encoding is independent of the resolution of the network and can be sampled to generate very large

[1] See Picbreeder.org and EndlessForms.com

network which, crucially, perform the same task as might have been evolved for a simpler case (smaller resolution). It is thought that this procedure may exploit the geometric regularities of the task domain.

II.1.2 Architectures for Control

In seeking to develop autonomous systems we must adopt a system of representation which encapsulates an appropriate range of dynamics with a degree of complexity sufficient to enable the desired behaviour without rendering synthesis intractable. Over the last few decades of Artificial Intelligence research a wide variety of architectures for autonomous systems have been introduced with many still occupying capability niches. For more specialised tasks, there may be a whole range of highly tailored options, but the search is still on for general solution capable of emergent autonomous and adaptive behaviour.

II.1.2.1 The Origins of Artificial Autonomous Systems

In the late 1980s, Brooks et al. pioneered a new approach to artificial intelligence, proposing a robust form of incrementally layered control system denoted the Subsumption Architecture. This approach advocated the design of independent layers of control, starting with base behaviours and adding complexity, with each system fighting for control of the robot, as shown in Figure II.1.1.

In their research, these control system layers were formed from a network of Finite State Machines (FSMs) connected together. Each module is a FSM which can hold Lisp data structures. Messages from modules may inhibit those of lower level control layers and thus gain control of the robot.

Brooks' [21] coordination of these parallel and distributed behaviours through activation levels roughly mirrors the biological hormonal control of

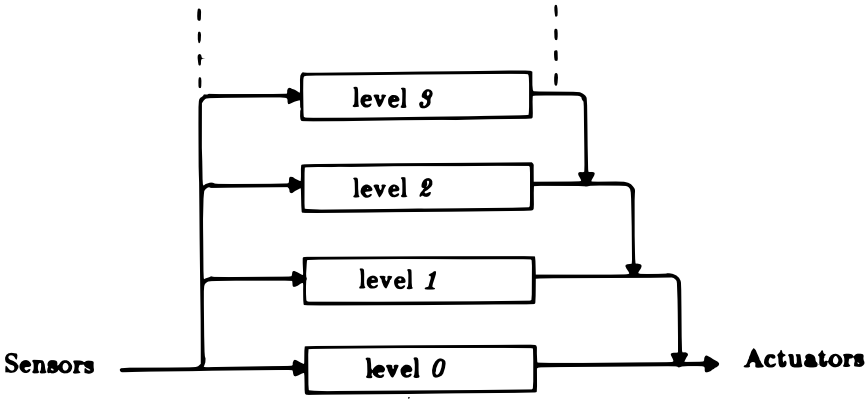


Figure II.1.1: Subsumption Architecture, from [18]

behaviour as described by Kravitz [67]. This architecture is interesting in its departure from GOFAI, and in the view that complex behaviour may emerge from the rich interaction of simple systems and dynamic environments. Whilst robust real robots were developed [23, 19], control systems for mobile robots are highly complex to design, and it becomes inherently more complex faster than the numbers of layers or modules in the architecture [57, 29].

There have been a wide variety of approaches explored for the development of autonomous control systems and it is beyond the scope of this review to cover them in more detail here. However, for the traditional design of cognitive architectures "it seems likely that the limits of feasibility for real robots doing useful things are currently being reached" [47, p.3].

II.1.2.2 Artificial Neural Networks

ANNs are a computational tool based upon a simplified neuron model of the brain. Nodes are connected to a selection of other nodes, and the output of any node is a function of its inputs. They have been proven to be *universal approximators*, in that an ANNs of sufficient nodes can approximate any arbitrary function, be it highly complex or non-linear [54], and that "a given

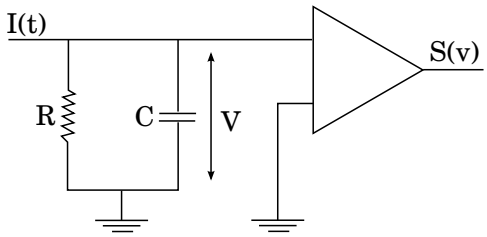
continuous mapping on a compact set can be approximately realized by three-layer feedforward neural networks with any precision" [38]. They are therefore tremendously powerful and usefully do not require large amounts of computation to determine the outputs.

However, as these networks can approximate anything, the difficult part is getting them to approximate the function that you require - particularly if you don't know what this is. The determination of the correct series of weights, thresholds and other parameters required for a given network is complex and may lead to very high dimensional solution spaces that would be impossible to characterise through an exhaustive search. GAs, simulated annealing and Monte Carlo methods, among others, have been employed as efficient search algorithms in these very large solution spaces. The vast majority of the literature in ERs has focussed on GAs, but many methods are potentially viable and simply serve as a way to find an approximation to the global optimum.

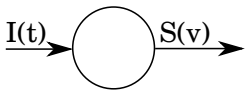
One reason for the popularity of ANNs in ER and Artificial Life is their inspiration taken from nature in which they clearly are capable of developing highly intelligent and adaptive behaviour typified by humanity.

> ... animals are endowed with nervous systems whose dynamics are such that, when coupled with the dynamics of their bodies and environments, these animals can engage in the patterns of behaviour necessary for their survival.[15, p.91]

Indeed, "There is little doubt that the brain uses active configurations of neurons to represent the properties of perceived entities and events." [8, p.582] as powerful, noise tolerant control systems with proven evolvability [47], ANNs are the architecture of choice for the study of ER. In biology, estimates place the computational speed of neurons to be very slow, at around 1 KHz but they appear to be highly connected. In small creatures



(a) Circuit Analogue



(b) Neuron Schematic

Figure II.1.2: A CTRNN node

individual neurons are connected to tens of percent of the total neurons in the body. In mammals, motor neurons are connected to around 5,000 other neurons, and some neurons in humans are connected to as many as 90,000 other neurons [22].

II.1.2.3 Recurrent Neural Networks

There are many types of ANN and they have been used in an incredibly wide range of areas, from pattern recognition and financial modelling to control. The first experiments in ER used simple feedforward networks, with later work focussing on more complex neuron models with richer dynamics capable of more interesting behaviour. CTRNNs have been shown to be able to approximate any dynamic system [38]. In particular they have been shown to exhibit useful properties, such as acting as oscillators [29], which is particularly useful in the evolution of gaits for walking robots and many other applications. Indeed, "Dynamic recurrent real-time networks form an extremely general class of control systems" [29, p.83].

The CTRNN model was created by Hopfield and Tank [53]. It is born out of an Resistor - Capacitor (RC) circuit analogue to a biological neuron,

aiming to retain the key dynamics but arrive at a simple model. The circuit analogue of a single neuron is shown in Figure II.1.2 where $I(t)$ is the current into neuron which is proportional to the average spiking frequency of the afferent neurons. $V(t)$ is the membrane potential, and the output of the node is $S(v) = 1/(1 + e^{-v})$. This maps the membrane potential of the differential equations to the output space. The derivation is as follows:

$$I(t) = \frac{V}{R} + C\frac{dV}{dt} \tag{II.1.1}$$

$$\frac{dV}{dt} = -\frac{V}{RC} + \frac{I(t)}{C} \tag{II.1.2}$$

$$\text{let} \quad \tau = RC \tag{II.1.3}$$

$$\therefore \quad \frac{dV}{dt} = -\frac{V}{\tau} + \frac{I(t)}{C} \tag{II.1.4}$$

$$\text{let} \quad y \equiv V \tag{II.1.5}$$

$$\tau\dot{y} = -y + RI(t) \tag{II.1.6}$$

$$\text{But} \quad I(t) = I^{int}(t) + I^{ext}(t) \tag{II.1.7}$$

$$I^{int} \propto \sum_{j}^{N}(w_{ij}\sigma(y_j - \theta_j)) \tag{II.1.8}$$

$$\text{Where} \quad \sigma(x) = \frac{1}{1 + e^{-x}} \tag{II.1.9}$$

$$\tau_i\dot{y}_i = -y_i + \sum_{j}^{N}(w_{ij}\sigma(y_j - \theta_j)) + I^{ext} \tag{II.1.10}$$

For the whole network, in vector format:

$$\boldsymbol{\tau}\dot{\mathbf{y}} = -\mathbf{y} + \sum_{j}^{N}(\mathbf{W}\sigma(\mathbf{y} - \boldsymbol{\theta})) + \mathbf{I^{ext}} \tag{II.1.11}$$

Where:

- $\boldsymbol{\tau}$ is a diagonal matrix of $N \times N$.
- $\mathbf{W}$ is a matrix of size $N \times N$.
- $\mathbf{y}$, $\boldsymbol{\theta}$, $\mathbf{I^{ext}}$ are external vectors of size N.

The specific form of neural architecture is vitality important to the performance and evolvability of the control system. This is covered in far greater detail in the rest of the review, but there is no doubt that "... circuit architecture does significantly constrain the maximal achievable fitness ..."[12, p.21]. This area of research is known as *Artificial neuroethology* which is the study of the relationship between artificial control network processes and behaviour [27].

Whilst initial experiments produced promising results, using highly simplified neuron models to evolve basic obstacle avoidance and goal seeking behaviour, Cliff et al. [29] argue that neuron models should be of a higher complexity than those used in a lot of the earlier literature on the evolution of ANN controllers, because:

> ... in using simplified models, we may be actually making life harder for ourselves as scientists; because the tasks we try to make our models perform may, but their very nature, require greater complexity than is possible without using clever 'trick' techniques, or large and unwieldy modular assemblies of simple networks. [29, p.40]

The authors point out that most models in the literature have limited dynamics and feedback, despite the fact that these feature are almost certainly what is required for most of the more interesting and challenging problems, as can be seen in how biological neural networks "exhibit rich dynamical behaviour and exploit feed-back connections to great effect" [29, p.40]. An extended form of genetic algorithm called SAGA was used in their research, which allowed the dimensionality of the parameter space to be under evolutionary control. The dynamics and structure of the neuron model used was of sufficient complexity that it was not necessary to alter the weights of connections; all weights and delays were fixed at unity. An innovative

encoding structure was used to allow variable length genotypes and arbitrary network structures without the defined layers of recurrent networks; care in crossover and mutation is essential to generate valid progeny in this method [47]. Controllers developed using this technique were shown to be fairly specifically tailored to the amount of internal noise present which is to be expected; more interestingly however, fitness decreased when evaluated with zero noise. This suggests that the networks were using the internal noise to avoid the effects of unproductive attractors [29].

Some work on including time delays within a neural architecture have been undertaken, but are limited in nature. Duro et al. used higher order synapses including trainable delays within a back propagation algorithm to pre-condition IR sensor data for a mobile robot controlled by a feedforward ANN [34, 16]. They argue that this approach allows the creation of virtual sensors where sensor data was temporally correlated to obtain a more useful representation of the work. The authors also applied their technique to Echo CardioGram (ECG) beat classification [33]. Pearlmutter addresses the issue of synapse delays in the context of back propagation training, but does not apply them [85]. Similarly, Lang and Waibel use a Time Delay Neural Network in word recognition [71].

Cohen et al. present the closest extension to a standard RNN to that which is developed in this Thesis, called the Time Delay Recurrent Neural Network (TDRNN). They claim that the inclusion of the delay may have advantages such as increased capacity resulting from the higher degree of freedom, reducing the network size and storing information without hidden neurons. Their model is of the form:

$$x_i = \sum_{j=1}^{N} w_{ij} y_j(t - \tau_{ij}) \tag{II.1.12}$$

$$T_i \dot{y}_i = -y_i + \sigma(x_i) + I_i \tag{II.1.13}$$

Where $I_i(t)$ is an external dynamical input function to neuron i, $\sigma(x)$

is the standard sigmoid function, $y_i(t)$ is the output of neuron i, T_i is the neuron time constant, w_{ij} is the weight matrix element connection neurons i and j, x_i represents the total input to neuron i, N is the number of neurons in the system, and τ_{ij} is the new delay parameter. They demonstrate the training of small networks for oscillations and pattern tracing.

II.1.2.4 Gene Regulatory Networks

Biological organisms without neural networks are still capable of sophisticated adaptive behaviour governed by regulatory networks comprising genes, proteins and small molecules [9]. Single celled Eukaryotic organisms exhibit fundamental behaviours such as motility and taxis; *E.coli* is the best known model organism for unicellular chemotaxis. The modelling and analysis of biological GRNs continues to be a very active area of research.

GRNs are interesting in that their behaviours range across a very wide range of timescales. Very fast protein interactions from ms to minutes, regulatory interactions between Deoxyribonucleic Acid (DNA) and Ribonucleic Acid (RNA) (where gene expression proteins are produced and degraded with a half life of minutes to days) from tens of minutes to days and epigenetic modifications from days to weeks to years.

> Natural regulatory networks are often very complicated, such that for even the simplest functions many components are involved and entangled with each other.[9, p.3]

A wide variety of techniques have been adopted for the synthesis, analysis and re-construction of GRNs from time series data, including coupled Ordinary Differential Equations (ODEs), Boolean networks, continuous networks, stochastic gene networks and higher order neural networks.

Beal et al. developed a biocompiler in which modelled GRNs are used to compile algorithms into genetic code which can be implanted in biolog-

ical entities [9]. In their approach each regulatory element is a functional unit consisting of a promoter, one or more genes, and a terminator. The gene is regulated, positively or negatively, by upstream elements whose concentration serves as the input signal. The promoter produces proteins as output that can serve as transcriptional regulatory factors inputs for downstream regulatory elements. The behaviour of each regulatory element is described by a multi-input sigmoidal transfer curve. The work here focusses on using the somewhat digital nature of sigmoid outputs to create "hybrid analog/digital circuits".

GRNs can be modelled as ODEs, but it is difficult to know what the initial conditions should be [93]. Biological systems have built-in regulation mechanisms which make them robust to changes in their parameters. Many Transcription Factors (TFs) form dimers or higher-order oligomers, so that their binding to DNA is sigmoidally dependent on their concentration. Purely Boolean representations have lower predictive power than their continuous counterparts (reviewed in [101]). GRNs have also been modelled as 'classical' or 'first order' neural networks [89, 117]. In the neural network approach, the genes form the nodes of the network. The connections between the nodes have weights, which relate to the magnitude of the effect that a gene at one end of the connection exerts on the gene at the other end. Weights can be zero (no interaction), positive (stimulation of gene expression), or negative (repression). The effects are assumed to be additive so that transcription levels are continuously valued. This approach ignores the fact that there is often significant synergism - defined as deviation from additive behaviour - in the effect of multiple TFs on the expression of a single gene. Thus, the classical ANN approach cannot be used to describe the ubiquitous 'AND' interaction, in which individual transcription factors have no effect, but their combination does. In an attempt to avoid the limitations associated with the Boolean and first order neural

network approaches, Yuh et al. used a combination of digital and analogue representations to model the regulation of the sea urchin embryo Endo16 gene [126, 127] . Genes are modelled as so-called Sigma-Pi units which were introduced as nodes in 'higher-order' neural networks by Rumelhart et al. [92], in order to circumvent linear separability constraints associated with first-order neural networks.

In a higher order network, a combination of inputs into the same node may generate an output that is different from their simple sum. Boolean functions and logic gates can be expressed in the Sigma-Pi formalism, but the input to and output of a Sigma-Pi function is not restricted to Boolean values [44]. Like logic gates, Sigma-Pi units are combinatorial, and consequently complex units may sometimes be decomposed into a set of simpler modules. Central to the approach advocated by Yuh et al. is, as in electronic design, a circuit diagram, which describes the connectivity and overall organisation of the network. The attraction is not only that the interactions between TFs are visualised in a simple, modular way, but also that the circuit diagram can be used as a basis for simulation models. However, the authors did not describe the precise relationship between TF binding, gene expression, and combinatorial or sequential logic, nor have the assumptions on which the relationship is based been made explicit.

Geard and Wiles developed a Dynamic Recurrent Gene Network (DGRN) and evaluated its ability to control developmental trajectories of cells during embryogenesis [39]. They hypothesise that the simple model possesses a sufficiently flexible range of dynamic behaviours to enable it to generate complex developmental trajectories. The DGRN follows a more abstract approach than other models [62, 102], focussing on insight to high-level properties. They model a network of N nodes, with activation states between 0 (inactive) and 1 (active), updated synchronously at time steps representing duration of cell divisions. N_s structural nodes (hidden), N_r regulatory

(output) nodes and single input node. $\sigma(x)$ is standard sigmoid function.

$$a_i(t+1) = \sigma(w_i I(t) + \sum_{j=1}^{N_r} w_{ij} a_j(t) - \theta_i) \qquad \text{(II.1.14)}$$

Fitness was measured as the error in node activations, $a \geq 0.5 \equiv$ Active and $a \leq 0.5 \equiv$ Inactive. This relied on *a priori* knowledge of the correct target pattern for the cell lineage tree of *C. elegans*. In the results presented the patterns were randomly generated or biologically inspired. The hidden structure of internal units was varied along with the size of the target patterns and their efficacy compared; see [62, 122] for analysis of the dynamics of such networks.

Knabe et al. adopted a continuous model, similar to higher order recurrent neural networks [64], when evolving GRNs to replicate behaviour of biological clocks given a periodic external stimulus. The stimuli had noise and blackouts added and their approach used shifted sigmoidal activation functions where the genome is represented as a binary string. Outputs were measured as to their proximity to a desired periodic output function with the aim of developing GRNs which are robust to changes in the environmental stimuli. Some solutions evolved which could function without external stimuli.

Li et al. focused on the model-free reconstruction of Time-Delayed Gene Regulatory Networks (TdGRN) from temporal gene expression data i.e. pairwise overlaps of expression levels shifted in time relative to each other [72]. They applied their approach to yeast cell cycling and human HeLa cell cycling. "Many important biological processes (e.g., cellular differentiation during development, ageing, disease aetiology etc.) are very unlikely controlled by a single gene instead by the underlying complex regulatory interactions between thousands of genes within a four-dimension space." [72, p.1]. The time-delayed gene regulation pattern in organisms is a common phenomenon [98, 124], so it can be conceived that multiple-time

delayed gene regulations are the norm and the single-time delayed ones are the exception. The exact time-delayed mechanism(s) remain to be verified experimentally however their data fitting approach matched the biological evidence as well as generating a number of new hypotheses.

GRNs for Robot Control

Robotic applications of GRN control are fairly limited and this is certainly a developing field. Despite the fact that biological neurons react on a faster timescale than genes this is no barrier in the computer modelling of such systems and as such present a viable architecture for control. Kumar developed a simple reactive obstacle avoidance controller for two different robots, the Pioneer 2DX and Active Media's Amigobot [69]. Similarly, Bentley demonstrated GRN controlled obstacle avoidance in a hexapod walking robot, where the GRN was not responsible for the gait but simply the navigation [17] and Zahadat et al. evolved a modular snake robot controller in a reactive task [128]. The work on obstacle avoidance behaviour was extended by Trefzer et al. in considering a range of different environments with a simulated and real E-Puck robot [113].

II.1.3 Dynamical Anaylsis

Further to the development and application of the models introduced in Section II.1.2, there has been a considerable effort involving the mathematical analysis and treatment of dynamics of such systems. It is fundamental that we understand these dynamics and how they may contribute to, and be best harnessed in, the development of autonomous systems.

There has been considerable, highly mathematical, treatment of RNN systems of a more general type than that used in ER (for example, see [55]). However, Beer was the first to consider the comprehensive analysis of sin-

gle and dual neuron CTRNN circuits [10]. Considering first autonomous circuits without time variant inputs, for a single neuron with and without self-connection he established the equilibrium surface and bifurcation set of the dynamics and then considered the phase portraits and bifurcation sets of a similar two neuron case. As networks increase beyond this complexity it is difficult or impossible to examine directly, but he proposed the decomposition of larger systems for the purpose of analysis as a combination of simple systems may be comprehensibly understood. In his later works, he further explored the local bifurcation manifolds and larger systems in an effort to develop a more general theory of dynamic neural circuits [11]. Through application of these results appropriate ranges of connection weights and thresholds can be selected for the richest dynamics.

This neurodynamical approach was applied to the analysis of the dynamics of brain-body-environment interaction for a hexapod walking robot [12]. By considering a single limb of the robot, it was possible to mathematically describe the fitness space structure of the system [13, 14]. The limitation of the problem for study to three dimensions allows powerful visualisation and analysis of the fitness space without resorting to indirect inferences using many concepts from [110]. This analysis "demonstrates some of the rich and subtle ways in which a neural and a mechanical system can interact" [13]. All work was carried out in simulation in networks without noise; the effect that noise would have on the controllers evolved and the analysis of the fitness space and dynamics remains to be seen. Indeed, the authors state that "It would be interesting to see how the picture developed here changes as we incrementally add more realistic dynamics to the body model" [13].

Others have further examined the dynamics of specific evolved agents to explore the mechanics of learning behaviour in a simple food discrimination task [86]. By strobing the dynamics of the system they were able to reconstruct finite state machines embedded within the neural network and

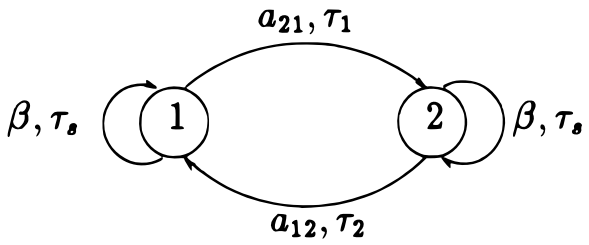


Figure II.1.3: Two coupled delayed neurons [96]

analyse their results.

Whilst the dynamics of Hopfield type neural networks with delays have been considered from a mathematical point of view - for existence, uniqueness and global stability of the equilibrium point have been undertaken [129] - this is done without any consideration of the dynamic behaviour or applications, let alone use in ER.

Shayer and Campbell considered the dynamics of two coupled Hopfield type neurons with variable delays but without thresholds or external input to the nodes as shown in Figure II.1.3 and Equation II.1.15 below [96]. The system was described by;

$$\dot{x}_j(t) = -\kappa x_j(t) + \beta \tanh(x_j(t - \tau_s)) \tag{II.1.15}$$

For the two neuron system they consider in Figure II.1.3, this becomes;

$$\dot{x}_1(t) = -\kappa x_1(t) + \beta \tanh(x_1(t - \tau_s)) + a_{12} \tanh(x_2(t - \tau_2)) \tag{II.1.16}$$

$$\dot{x}_2(t) = -\kappa x_2(t) + \beta \tanh(x_2(t - \tau_s)) + a_{21} \tanh(x_1(t - \tau_1)) \tag{II.1.17}$$

Their analysis is interesting, but does not consider any external input to the network, has no threshold and uses the hyperbolic tangent as opposed to the sigmoid function for node activations.[2] They attempt to describe the largest subset of a five-dimensional parameter space in β,κ, γ (a function of the weights), τ_s and τ.

[2]The two are almost identical, but *tanh* tends asympotically to -1 or +1, whereas the sigmoid function is bound between 0 and +1.

The study of GRNs with time delays is more advanced with considerable mathematical theory on their stability [28, 84, 118]. Wang et al. [119] undertook a complex mathematical analysis of stability in delayed GRNs, modelled continuously by differential equations. They state that:[3]

> It has been recognized that the slow processes of transcription, translation, and translocation or the finite switching speed of amplifiers will inevitably cause time delays, which should be taken into account in the biological systems or artificial genetic networks in order to have more accurate models [77]. It has been shown [82], by mathematically modelling recent data, that the observed oscillatory expression and activity of three proteins is most likely to be driven by transcriptional delays, and delays can have significant impact both on the dynamical behavior [sic] of models and on numerical parameter prediction. [119, p.154]

When the gene network is viewed as a dynamical system, external control (or intervention) can be applied to make the controlled network achieve desired behaviours such as the stability of the gene expression level regardless of the parameter variation within a certain range. Time delays are frequently encountered in not only the biological networks but also many other practical engineering systems, such as communication, electronics, hydraulic, and chemical systems. It is now well known that time delay is one of the main causes of instability and poor performance of a control system [78].

II.1.4 Evolution of Adaptive Behaviour

By coupling the control representations introduced in Section II.1.2 and the methods for Artificial Evolution described in Section II.1.1 it is possi-

[3]References within the quote have been modified to be correct within the content of this Thesis.

ble to explore the synthesis of autonomous systems capable of learning and adaptive behaviour. This interest may be driven by the desire to better understand the underlying mechanisms of the biological world, or to investigate the emergence of intelligence in artificial systems and robotics control. The latter is the field of Evolutionary Robotics which approaches the same subject with more of an engineering approach than that of the biologist [45].

II.1.4.1 Embodied Intelligence

Mobile robots are intrinsically embodied in a challenging, dynamic environment and there is a rich, deep relationship between the control system and the morphology of the robot, presenting both challenges and opportunities for evolutionary robotics. In fact, "behaviour is best viewed as a property of a complete brain-body-environment system [...], and cannot be properly assigned to any individual component of this coupled system" [12, p.8].

There is a significant body of evidence for the embodiment of artificial intelligence and the effect that the morphology and structure of the 'body' has on the control system. Brooks [20] argues vehemently that grounding of robots is essential for the development of higher level behaviours, in contrast to the view of traditional artificial intelligence symbolic representation based research. Mobile robots live in the real world, extracting data from sensors and influencing their environment with their effectors. In this way the form of the robot and the arrangement and structure of its sensors and effectors are of fundamental importance, and will have a most significant impact on the form, function, evolvability and capability of any evolved controller. Varela [115] states that:

> By using the term embodied we mean to highlight two points: first, that cognition depends upon the kinds of experience that come from having a body with various sensorimotor capacities, and second, that these individual sensorimotor capacities are

> themselves embedded in a more encompassing biological, psychological and cultural context. By using the term action we mean to emphasize once again that sensory and motor processes, perception and action, are fundamentally inseparable in lived cognition. Indeed, the two are not merely contingently linked in individuals they have also evolved together. [115, p.172-173]

As reported by Brooks [24], as much as 50% of the human brain appears to be dedicated to perception. This is perhaps unsurprising, as it is through our sensing of the environment that we are able to make decisions, gain understanding and act to influence that environment. So, whilst is it apparent that robots must have sensors to be able to probe the environment to gain any degree of autonomy or usefulness, it not so clear as to which sensors and in what arrangements. Sensors may be classified as proximal or distal, interoceptors or exteroceptors and their specification has an immense impact on the form, efficacy and evolvability of controllers.

Cliff et al. argue that vision is essential for interesting robots, but that with high resolution / high bandwidth devices there are tremendous issues in the genotypical encoding and simulation, and so most investigations thus far have used very low resolution visual devices (mostly photoreceptors). This is biologically justified and ties in with the sensory apparatus of many insect species. Experiments reported by Cliff et al. illustrated that once individuals learnt to use the visual sensors, the other sensory inputs fell into disuse and were reallocated as further visual processing units which resulted in a reduced neural structure [29].

By choosing the right sensors, animals can often get by with very little neurological processing, just enough to extract the required information for the task at hand. The building of complex world models does not appear to be neurologically supported in biological systems, contrasting with large bodies of work with traditional artificial intelligence [22]. Wehner [120] illus-

trates a number of examples focusing on natural vision systems. Particularly relevant is the *Skylight Compass* that bees use to localise themselves based on the linear polarisation of light from the sun. The analytical solution to this is complex, three-dimensional and requires significant optical knowledge, but:

> Are we really to assume that the insect's brain comes programmed with the ability to perform such three-dimensional constructions in the sky, let alone to acquire, by either evolutionary or individual experience, the underlying knowledge about skylight optics? I doubt it ... [120]

The biological solution is much simpler than a physicist's approach, and instead uses *matched neural filters* consisting of an array of polarization sensitive photoreceptors whose direction is matched to the polarization pattern in the sky, requiring a simple biological analysis to generate the required information. This 'compass' measures relative rather than absolute intensities making it robust against intensity fluctuations due to outside influences. Thus evolution is shown to exploit 'tricks' to generate simple, but specific solutions to problems in an entirely different manner that a human designer would approach the problem. Wehner [120] concludes by saying;

> The systems are not perfect, but they work sufficiently well not to be pared down by natural selection. When there is no need for a more perfect solution, why bother with it? This is certainly the elegant way of solving a problem - and the brain depends on elegance to compensate for its small size and short lifetime [120, p.530]

Wehner [121] provides evidence from experiments on bees that contradict previous suggestions that bees practised map based navigation. The

experiments showed that the insects clearly followed skylight compass based headings and used natural optometry to navigate in combination with landmark recognition. When approaching a goal the bees aim to minimise differences between their current retinal patterns and learned patterns. Whilst this takes place on at least partially processed representations, there is no evidence to suggest integration into a complex map form.

> However, an artificial agent does not necessarily need to integrate different senses into a coherent picture, since many successful predators in the (natural) animal kingdom do not do this either. A snake, for example, employs them sequentially ... [90, p.357]

Harvey et al. [47] argue that in order to develop more sophisticated navigational competences than blind meanderings, an increased reliance on *distal* sensors, particularly vision, is necessary. The fusion of visual information into neural networks for high resolution images leads to networks of great complexity and size outside the current scope and feasibility of ER research, but:

> That many animals, particularly insects, successfully occupy their ecological niches using low-resolution low-bandwidth vision as a primary source of exteroception information indicates that such an approach (as opposed to high resolution and bandwidth) is worth exploring within an evolutionary robotics context, in the first instance at least [47, p.6]

Alongside exteroceptors to sense the external world, "Truely autonomous robots will require interoceptors to monitor significant internal states such as power level and degree of wear or damage" [29]. The implementation and representation of sensors and effectors within the evolutionary process is vital for the success of evolved agents in reality, and their adaptive capabilities.

II.1.4.2 The Reality Gap[4]

Transferring controllers from simulation to physical robots is a major challenge. The use of simulations was criticised by Brooks [24], warning of the danger that controllers evolved purely in simulation are highly likely to fail due to the complexities of real life. More recent work by Jakobi [60] shows how, through the appropriate introduction of noise into simulations, successful robust controllers which behave in the real world as they did in simulation may be obtained. It is recognised however that it is probably far easier to construct accurate simulations of the simple Khepera robots used than for many others. Simulations were spatially continuous and two dimensional, using empirical information, thus contrasting with the view of Brooks that any model must be 3D to accurately simulated interactions in the real world. A distributed GA was used to avoid issues of premature convergence. Experiments were carried out with no noise, observed noise and twice the observed noise to quantify its effect. In the course of this research there was never a binary negative correlation between observed and expected robot behaviour. The conclusions presented indicate an inverse relationship between the level of noise included in the simulation and that observed on the real robot. How did the noise affect convergence time for the evolutionary runs? Whilst it is intuitive to realise that the evolutionary process will take advantage of the zero-noise case, it is less obvious that the same occurs for the double-noise case. This may result in significantly poorer evolved individuals when transferred from simulation to reality, and so it is crucial to include the correct amount of noise. Whilst the authors report concern over the increasing level of complexity of interaction dynamics for more complex systems, they conclude that simulations may still be useful within this area. An alternative approach is to evolve the robots in real time on board hardware, following the philosophy of Brooks that the world is its own best model. However, it

[4]A term coined by Jakobi [60]

must be considered that in both cases it is necessary for the environment to be fairly specific for the successful evolution of fit robotic controllers. The generalisation of these controllers so that they may work well in different environments has yet to be satisfactorily addressed.

> Clearly, the main challenge for the simulation approach to evolutionary robotics is to invent a general theoretical basis and methodology on which fast-running simulators can be easily and cheaply built that guarantee the transference of evolved behaviors [sic] from simulation to reality. [59, p.326]

Further to the work reported in [60], Jakobi [59] proposes a *base set* of robot-environment interactions that must be included in simulations to successfully transfer the evolved results into reality. The *Radical Noise-Envelope Hypothesis* is proposed whereby simulations should include the base-set of interactions, and every implementation aspect of the simulation should be randomly varied so that controllers who rely on them are unreliable, so that "... *enough* variation must be included to ensure that evolving controllers cannot, in practice, be reliably fit unless they are base set exclusive" [59, p.332]. Also, from trial to trial, the base-set aspects should be varied to induce controller robustness to these factors which is essential to enable the controller to soak up the differences between simulation and reality. However:

> There is a real danger, if we are overzealous in our lust for computational expediency, that we may effectively exclude so many real-world features from the simulation that what is left is insufficient for successful behavior [sic]. [59, p.334]

Jakobi successfully generated minimal simulations, which had little obvious connection to the physical problem, and produced robust, reliable

individuals which successfully crossed the 'reality gap'. The computational abstraction and simplifications meant that in one instance 3 years of simulated time was processed in 12 hours on a single computer. The exponential increase in computational power over the years would no doubt have made this comparison even more startling. Jakobi demonstrated the power of such minimal schemes to develop 'complex' behaviours, whilst through careful design of simulations following three simple principles successful transfer across the reality gap may be assured. To conclude, Jakobi noted that:

> The point is that whether a minimal simulation is easy to construct and runs fast depends not on the complexity of the behavior [sic]we want to evolve when using it, nor on the complexity of the robot that it simulates, but only on the complexity of the base set of robot-environment interactions necessary to underlie the behavior [sic]. Provided these are simple enough, then the behavior [sic] or the robot (or both) can be arbitrarily complex. [59, p.365]

Lund et al. [75] argue that the time scale required to obtain scientifically valid (repeated) results with the physical embodiment of the evolutionary process is infeasible. Their approach is to develop controllers in simulation, tailored for phenotypic plasticity to soak up differences, and then complete the process through on-line evolution. The accurate definition of simulations may also be time consuming because of the need to sample of all possible sensor readings for non-trivial environments and morphologies. The researchers reported that a sample based look-up table simulation took three times longer. However, the sampling of the environment using the physical robot assists in building up a physically realistic simulation and reduces the transition between simulation and real life.

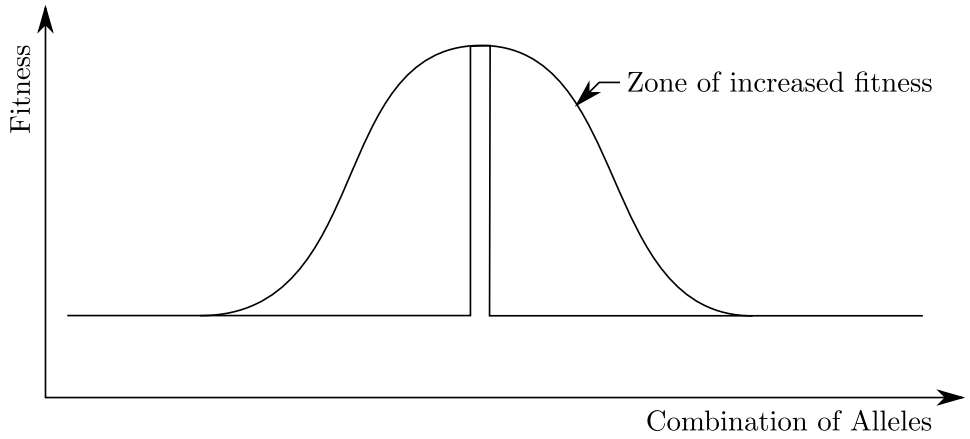


Figure II.1.4: How learning affects evolution, reproduced from [49]

II.1.4.3 Evolved Learning

According to Baldwin [4], learning accelerates evolution because sub-optimal individuals can reproduce by acquiring during life necessary features for survival, so learning affects evolution, despite the fact that learning is not genetically expressed in the population. Hinton et al. [49] developed a highly simplified computer model to show how learning might help and guide evolution. They found that learning tends to smooth the fitness landscape to make evolution more efficient by allowing near fit individuals to learn, gaining fitness, during their lifetime as shown in Figure II.1.4.

Continuous real-valued artificial neural networks with time delays and unrestricted topologies are very powerful for the evolution of control systems, but they generally have a high degree of design complexity compared to the 'optimum' solution to previously studied problems [58].

Yamauchi and Beer [123] eschewed the sharp division of sequential and learning behaviours in autonomous agents and attempted to evolve dynamic neural networks to generate this behaviour without the aid of an external learning algorithm (e.g. modifying network weights within the life time of an agent). They applied this approach to landmark recognition, 1-D navigation

and sequence learning using CTRNNs.

They successfully evolved a simple eight node CTRNN for the landmark recognition on a mobile robot, whilst a simple behaviour based control system circled the landmark to be recognised. However, an integrated network for the solution of the 1-D navigation task was unsuccessful, and so an incremental modular network was evolved, with three sub-networks evolved for specific decomposed tasks. The classifier sub-network had a specific input neuron for a reinforcement signal, which was received for 50 time steps on the successful completion of the task. This approach was successful, but relied on the decomposition of the control network, and little attempt to justify this was made. Again, in the sequential learning task, the same problem was encountered, that an integrated network could not be evolved to solve the task, but an incremental approach was successful. This approach to RL differs significantly from the traditional approaches [111]. They concluded that the limitations of such a structure were not in the networks, but the time required for GA search of scaled problems, but provided no justification for the failure of the integrated approach.

Tuci et al. [114] further the work of Yamauchi and Beer by attempting to evolve an *integrated* CTRNN for the one-dimensional learning task, and investigate the failure of integrated networks in [123]. They argue that CTRNNs should be able to simulate this form of learning, calling on the work of Funahashi and Nakamura [38], and suggest that the nature of the constraints imposed in Yamauchi and Beer's model prevented their successful evolution. In contrast to their work, there was no dedicated reinforcement sensor input nor explicit reinforcement signal, thus bringing the problem closer to that of biology - what they term the 'ecological approach'. The 1-D model used by Yamauchi and Beer is replaced by a 2-D mobile robot task of the same structure, and a controller was evolved for a Khepera miniature robot. The robot was allowed to exploit richer sensory-motor capabilities

that the original model, and the landmark was visible throughout the whole arena in contrast. Therefore, whilst the search space was significantly larger (a 13 neuron CTRNN), the task was potentially easier to solve. Evolution took place in a deliberately noisy simulation, and the fitness was determined by the robot's success at the task, with penalties for collisions or incorrect moves. This evaluation function is incremental, designed to allow the agent to recover from mistakes, unlike the binary approach of Yamauchi and Beer. Despite differences to the experiment that were intended to make it easier to solve, it was unsuccessful. The experiment was modified to bias the relative weighting of the scores achieved in each environment, to encourage the robot to utilise the light source which the initial populations failed to do. This modification generated a majority of highly fit individuals that would learn from their mistakes and quickly adapt their behaviour to find the goal.

> ... if an organism is not able to recognise a cue and distinguish it from other environmental stimuli, it will be unable to form an association between the cue and an environmental state. [114, p.217]

Therefore, by biasing the environment so that it was advantageous for even non-learning robots (i.e. those capable of a reactive phototaxis strategy), agents evolved to 'pay attention to' the learning cue before it had any significance as such. Interestingly, in contrast to Yamauchi and Beer's model [123], there was no dedicated reinforcement signal, instead they "require that agents must evolve their own conditions of reinforcement, relying on existing sensors" [114, p.207].

Despite differences between the models used in [114] and [123], both investigations initially failed to evolve controllers to perform an associative learning task. By biasing the test placements, Tuci et al. ensured that the robots evolved to take account of a light source before then associating it with the navigation task, resulting in successfully evolved learning.

Bullinaria [26] used the evolution of learning agents to explore the effect of life time learning on protection periods. Learning was implemented through gradient weight updates of a simple multi-layer perception ANN. Within humans and other animal species there are "*critical periods* for learning, and outside that period the learning is more difficult" [26, p.391]. Whilst this research focuses on the Artificial Life approach of using evolved systems to study the natural world, the results suggest that parental protection and life time learning may be an important factor in the evolution of complex learning agents.

> Learning is also crucial for humans, and for other species for which relatively complex behaviors [sic] are required, since encoding all the necessary skills genetically is likely to be difficult, and even if that were evolutionarily possible, adaptation would still be needed to cope with their rapid growth processes and the changing and unpredictable nature of their environment. [26, p.390]

When considering learning in ANNs, it is worth noting that: "This is no short-cut recipe, but requires that the internal complexity of the 'brain' (of an organism or a machine) be dependent on the history of interactions with its world; the more the complexity that is required, the longer the history that is needed to mould it" [47, p.5]. So, for more complex controllers it may be necessary to study their behaviour and ontology for protracted periods to fully realise their potential. This has the potential for a significant impact on the required evolutionary strategy, not least on the amount of computation.

Maniadakis et al. [79] used CTRNNs to investigate cognitive processes involved in meta-rules, i.e. rules for rules. These potentially provide a basis for higher level evolved behavioural selection which is highly contextual in its nature. Bottleneck and fully connected CTRNNs were used, the former to separate the upper and lower parts of the network, segregating information

processing into two different levels. Their architecture was designed to model some the basic characteristics of cortical organisation:

> In the mammalian brain, it is well known that the reward information is projecting [...] to the prefrontal cortex, which is a module with higher level cognitive responsibilities, while other somatosensory modalities are directly connected to lower level motor modules such as primary motor cortex. [79, p.63]

In this case, the behavioural task was based on the Wisconsin Card Sorting (WCS) test, but implemented on a mobile robot (using a Khepera simulator) considering "both meta-level cognition and sensorimotor coupling as inseparable parts of a complex behavioural problem" [79, p.64]. The simulated robot has to decide to turn left or right at a T-junction based on a light cue. If the robot makes the wrong decision it is punished, and from time to time (unknown to the robot) the experimenter switches the punishment rule. The neural state of the controller was not reset, and thus was continuous across trials. Incremental evolution was successfully applied to a population of CTRNN controllers using a Genetic Algorithm. Specifically, the number of switch phases was increased across segments of the evolutionary run. The bottleneck CTRNN was shown to be significantly more successful at generating successful controllers (8 out of 10 runs, compared to 3 out of 10 runs for the fully connected model). Neurons at the higher level were shown to develop a two-state activation level determining which of the evolved behaviours to follow. Indeed:

> It is important to note that the development of rule-correlated dynamical states has been observed in the CTRNN of all successful evolutionary runs, implying that attractor dynamics might be an important general mechanism for manipulating rules in continuous time systems. [79, p.70]

For a more complex three phase switching task, significantly larger populations were required and even then, only three out of ten bottleneck evolutionary runs were successful. For this form of meta-cognitive behaviour it appears that the bottleneck topology is far more suited that the standard fully connected model, possibly providing evidence for the higher cognitive roles of the pre-frontal lobe in humans. Previous studies by the authors examined positive rewards, which were found to give similar results in the self-organisation of attractor dynamics.

II.1.5 Locomotory Robotics

Other than adaptive behaviour, there is one highly popular domain for ER: the development of controller for legged robots. This is because legged robots may be able to traverse rugged and otherwise impassible terrain, yet hand designing optimal gaits is challenging and time-consuming [125].

Beer and Gallagher evolved standard CTRNNs for both wheel robot chemotaxis and hexapod locomotion [15]. Their locomotion model is effectively one dimensional with legs simply adopting an up or down stance. The robot was only allowed to move when statically stable, but no other dynamical considerations were implemented. Whilst the individual neurons are not capable of oscillatory behaviour, they successfully evolved gaits with and without sensory feedback, thus in effect producing Central Pattern Generator (CPG). They claim that CPGs have a disadvantage in that they are incapable of taking advantage of sensory information when it available so are less flexible. Chapter II.2 illustrates how the approach advocated in this Thesis allows for single neurons to not only behave as pattern generators but also receive and process external stimuli.

Seys and Beer adopted the highly simplified model of hexapod locomotion of [15] and control it with a CTRNN without sensory feedback for the investigation of symmetry and genotype reuse in the evolution of gaits

[94, 95]. Later, a range of models were evaluated and numerically analysed [12].

In his Thesis, Ghigliazza investigated a wide range of neuromechanical models for insect locomotion, including neural networks and CPGs, with a fairly minimal leg model [40, 41]. Along with Kukillaya et al. simplified mathematical models were used to analyse the non-linear dynamics of the systems in an effort to find the underlying mechanisms of biological insect locomotion [68]. This is assimilated within the excellent review of Holmes et al. which covers a wide range of simplified models and techniques [52].

In investigating open-ended evolution of morphology and control of locomotory organisms Lipson developed controllers for a variety of forms [74]. A mix of simulation and embodied evolution was employed, and results obtained for quadrupeds, hexapods (the robot actually had nine legs, but only six could contact the ground at any one time) and bipeds. Influenced by the pioneering work of Sims [100, 99] evolved solutions were rapid prototyped and evaluated.

MacLeod et al. developed an incremental growth approach to generating modular neural networks and morphologies [76]. The process started with a basic 'mudskipper' with one active Degree of Freedom (DOF) legs and developed into a quadruped robot with vision. All work was carried out in a simulation environment and the neuron model was a 'Spike Accumulation and Delta-Modulation' form.

Quadruped robots controlled by CTRNNs in a 3D virtual environment were used by Auerbach and Bongard in their investigation as to how morphology and training order affected the learning of multiple behaviours [3]. The robots had grippers and were controlled by 11 motor neurons with the simulation and network updated every 0.0005s.

Clune et al. evolved gaits for a quadruped robot using HyperNEAT and the ODE [30]. The networks consisted of three 2D 5x4 Cartesian grids

forming an input, hidden and output layer with 800 non-recurrent links. Inputs to the network included joint angles, touch sensors, and a modified driving sine wave which allowed for periodic behaviours. Similarly, Yosinski et al. used the same techniques but in hardware on a quadruped robot [125].

II.1.6 Conclusions

This Chapter has covered a wide range of interrelated areas of research from computer science, biology and even aspects of psychology in an effort to place the results of this Thesis in context and establish the areas of novelty covered within.

Any foray into Evolutionary Robotics must apply some form of Genetic Algorithm stemming from the pioneering work of Holland [51]. A range of algorithms specialised for a variety of purposes have been introduced but perhaps focussing on the recent development of the NEAT methodology of Stanley [104]. The complexification approach by which a minimal genotype develops has been shown to be highly efficient, and in combination with the HyperNEAT algorithm [106] for evolved pattern producing networks [105], is highly suited to the work presented herein.

Since the birth of ER as a field of research, Artificial Neural Networks have been employed as the representation of choice [83]. Significant attention has been focussed on the CTRNN, developed by Hopfield as an analogue to a biological neuron [53], as it is capable of emergent adaptive behaviour beyond that of the discrete model. More recent interest has been lavished upon Gene Regulatory Networks (GRNs) which are thought to be at the core of most biological systems. Indeed in simple systems, such as unicellular Eukaryotes, they are wholly responsible for the adaptive behaviour exhibited. Whilst they can be modelled using a variety of techniques we have concentrated on the Ordinary Differential Equation (ODE) approach, which in some cases explicitly uses higher order continuous neural network

models. GRN dynamics are governed over a wide range of dynamics and are typically slower in acting than neurons. Temporal dynamics are therefore highly important in capturing their behaviour, and delayed systems of the form advocated in this Thesis are common. Despite this, robotic applications of delayed neural networks have thus far been limited to the preconditioning of sensor data and the majority of work limited to traditional back-propagation training and functional approximation.

Dynamical analysis of these systems is well established with research being both purely mathematical and within the context of ER as a tool to aid understanding and guide the development of interesting solutions. Analysis of delayed systems is limited to their stability and does not consider the characterisation of oscillatory behaviours. This allows even single delayed nodes with a recurrent connection to behave as Central Pattern Generators (CPGs) which would otherwise require larger systems of CTRNN neurons.

There are essentially two distinct advantages hypothesised for the inclusion of time delays in these systems. The delay enhances the information processing and storage capability of the synapses in the network allowing for smaller systems to develop complex behaviours. In addition, the increased richness of the dynamics may be harnessed for the easy development and inclusion of pattern generating components within networks which may present advantages across a wide range of potential applications.

Some fundamental aspects of ER have been reviewed, namely the use of minimal cognitive models, the reality gap from simulation to hardware and the evolution of learning in autonomous systems. Time and again biological examples illustrate how even very small neural systems can solve complex problems with elegant simplicity e.g. the skylight compass [120]. It is in this manner that we hope to study and develop incredibly small neurological circuits capable of interesting behaviour. However interesting it may be to use these techniques for the understanding of cognition, our focus is that of

the engineer in implementing them for the development of real autonomous systems. This required the transfer of systems most often developed virtually into hardware with the difficulties that this presents. A brief survey of the literature involved in the evolution of learning and adaptive behaviour is presented to place the contents of this Thesis in context and illustrate commonly studied examples in the field. Finally, the common application of ER techniques to the development of gaits for legged robots was introduced and relevant works presented.

This presents an opportunity to explore the fascinating dynamics of delay systems with an ER perspective and apply the increased capability of such systems to the evolution of gaits and adaptive behaviour which is the subject of the remainder of this Thesis.

Chapter II.2

Dynamics of Small Delayed Dynamic Networks

This Chapter investigates the dynamics of small time-delay networks and compares them to that of the CTRNN model, which is a close relation. Without an in-depth understanding of the changes brought about by introducing a simple model of transport delays in network connections, it is impossible to both apply such systems to greatest effect or properly analyse results of any investigation and so answer the research questions posed in Part I. The analysis presented here is both analytical and numerical, as such systems are highly complex in their nature and entirely analytical investigation may be impossible. For a single node of a time-delay network the behaviour has been exhaustively investigated and analysed, building a greater understanding of such systems and how they can be used together in more complex networks. The differences in behaviour that have been introduced are exploited in latter parts of the Thesis, notably in inspiring the application of delayed networks to robot locomotion in Chapter III.3 and the development of supporting material in Chapters II.3 and II.4. That said, the conclusions of this Chapter are used throughout Part III in analysing the results of every experiment.

II.2.1 Synaptic Time Delays

The increased dynamic complexity of the Hopfield-type neuron model [53], as used in CTRNNs, may be exploited by artificial evolution to develop controllers capable of going beyond reactive behaviour [45]. In biological systems, however, neural signals take time to propagate along synapses, providing another aspect of neuronal dynamics not modelled by standard CTRNNs, but a simple modification to this neural model may include these dynamics.

Incorporation of synaptic time delays is inspired by those inherent in biological networks which vary over a range of several orders of magnitude and play a significant role in their behaviour [31]. These additional time dynamics may therefore provide extended behavioural possibilities as the delay parameter increases the effective dimensionality of the system [70]. However, the additional evolutionary parameters required will potentially render the evolution of systems more complex and computationally expensive.

RNNs with signal propagation delays along synapses are a subset of DDE, shown in Equations II.2.1 and II.2.2 [70].

$$\dot{X} = F(t, X(t), X(t - \tau_i)) \tag{II.2.1}$$

$$X = (x_1(t), x_2(t), \ldots, x_n(t))^T \tag{II.2.2}$$

Where $\tau_i > 0$, $i = 1, 2, \ldots$, are lag times or delay times and F is a vector valued smooth continuous function. The delays may be constant, discrete, distributed, state-dependent or time-dependent. The trajectory of this Equation is dependent not only on the current value of $x(t)$, but on values at earlier times, $x(t')$, $t' \in (-\tau, 0)$. The time-dependent solution of the system must have a defined solution profile (initial function) for an interval set by the longest time delay. This requires a set of infinite (but continuous) initial conditions for $-\tau < t < 0$, hence DDEs are effectively infinite dimensional systems even with only a single delay [70].

The model introduced here combines the delay modelling of the Time Delay Recurrent Neural Network (TDRNN), as used by Pearlmutter [85], with the form of the much studied CTRNN. Here it shall be referred to as a Continuous Delayed Recurrent Neural Network (CDRNN). The output of the network is given by a system of ODEs shown in Equation II.2.3. The new parameter τ_{ij} represents the continuous delay between neurons i and j.

$$T_i\dot{y}_i(t) = -y_i(t) + \sum_{j=1}^{N} w_{ij}\sigma(y_j(t-\tau_{ij}) + \theta_j) + I_i(t) \qquad \text{(II.2.3)}$$

Where $I_i(t)$ is an external input to neuron i, $\sigma(x)$ is the neural response function, defined by $\sigma(x_i) = 1/(1+\exp^{-x_i})$, $y_i(t)$ is the output of neuron i, T_i is a time constant of each neuron, w_{ij} is the weight of the connection between neurons i and j and N represents the number of neurons in the system. Both τ_{ij} and T_i are measured in seconds. The motor neuron activities are mapped into the output space by $o_i = \sigma(y_i + \theta_i)$.

For a non-linear dynamic system without delay at least three dimensions are required for chaos to manifest, whereas even a single DDE can exhibit both chaotic and hyperchaotic behaviour even for small delays. There is a great deal of literature on the analysis of DDE systems typically approached from a classical control point of view or in terms of signal processing [33]. Shayer et al. considered the dynamics of a small neural network of Hopfield type with variable delays, but without thresholds or external input to the nodes [96]. Around a decade ago, researchers utilised discrete Mult-Layer Perceptron (MLP) networks with delays in robotics, but with a focus on sensing and purely reactive behaviour [34, 16]. The synaptic delays in discrete networks were used as an alternative to continuous recurrent forms, and were used to improve the precision of object recognition using IR sensors. They suggest that, "There is no reason why the explicit management of time performed by delay based networks should not be combined with recurrences or any other structure that allows the cognition mechanism to

obtain different perspectives of the information present in the environment" [16]. In this Chapter, the previous work is extended with a view to enhancing the learning and temporally adaptive behaviour of behaviour based robotics utlising delayed recurrent neural networks.

We can be confident that these CDRNN networks are capable of solutions to problems demonstrated by CTRNNs, as the evolutionary mechanism may reduce the time delays to zero and so model a standard CTRNN. However, it is true that the additional parameters may render the evolutionary process more difficult. When the inputs are time variant, as in any real system, the synaptic delays may considerably alter the outputs of the system. An example of this is shown in Figure II.2.1, where the outputs of three different neuron models (ANN, CTRNN and CDRNN) for a randomly generated neural network are shown for a step input halfway through the simulation. The network consisted of 3 input nodes, 3 hidden nodes and a single output node; all of the network parameters were randomly generated but are consistent between neuron implementations where applicable. Thus, the CTRNN implementation is identical to the CDRNN without the addition of the time delays, so any differences in network output are solely due to the synaptic delays. The CDRNN behaves very differently to the CTRNN despite their fundamental similarities, with an output profile which is typically more complex than that of its close relative, but converges to the same steady state.

Whilst this neuron model is continuous and governed by a system of differential equations, for implementation on any digital system (be it for simulation or real-time control) these equations must be numerically integrated. This requires the evaluation of Equation II.2.3 and that the outputs of each node be stored and delayed by a period determined by the time delay of each synapse which is covered in detail in Section II.2.4.

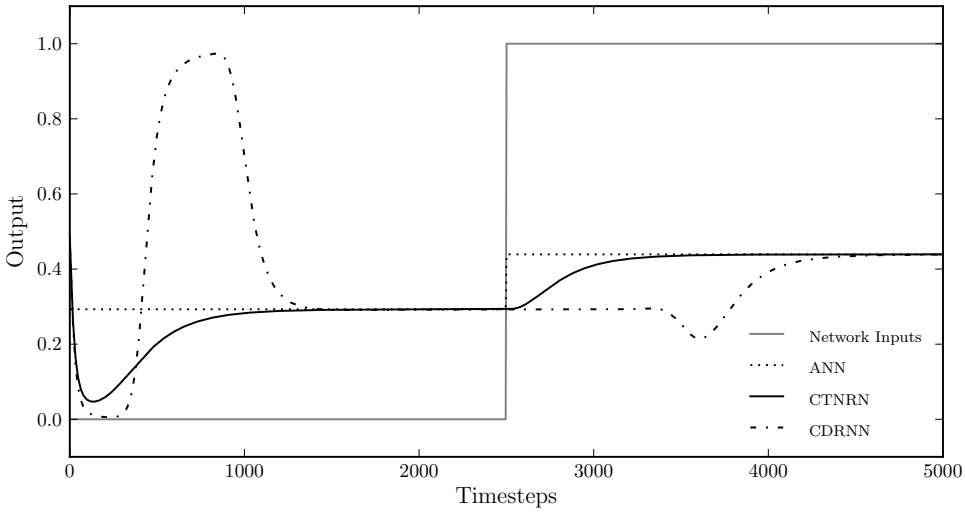


Figure II.2.1: Time Delay Dynamics in a Random Network

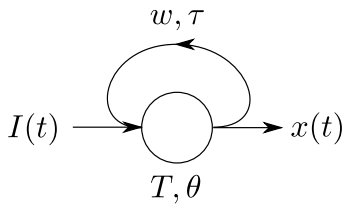


Figure II.2.2: A Single Neuron System

II.2.2 A Single Neuron System

Specifically, we are interested in an extension of the standard CTRNN [47] by the inclusion of time-delays, as shown in Equation II.2.3. The natural starting point for any dynamic analysis of the behaviour of such a system is to consider a single node of a network as shown in Figure II.2.2. This is because it is the fundamental building block of which these systems are formed. The behaviour of even small systems of equations can be incredibly complex and well beyond a symbolic analysis even without the consideration of time-invariant inputs to the network. The governing equation for a single node is therefore a single non-linear DDE of the form:

$$\dot{x}(t) = \frac{1}{T}\left(-x(t) + w\sigma(x(t-\tau)+\theta) + I(t)\right) \tag{II.2.4}$$

Were we to consider a two node system the governing DDEs would be:

$$\dot{x}_1(t) = \frac{1}{T_1}\left(-x_1(t) + w_{11}\sigma(x_1(t-\tau_{11}) + \theta_1) + w_{21}\sigma(x_2(t-\tau_{21}) + \theta_2) + I_1(t)\right) \tag{II.2.5}$$

$$\dot{x}_2(t) = \frac{1}{T_2}\left(-x_2(t) + w_{22}\sigma(x_2(t-\tau_{22}) + \theta_1) + w_{12}\sigma(x_1(t-\tau_{12}) + \theta_2) + I_2(t)\right) \tag{II.2.6}$$

This system of equations is said to be of advanced type DDE, but should any of the time delays tend to zero then it becomes a neutral type DDE.

The governing equation for a single node is given in Equation II.2.4. However, the same qualitative behaviour is exhibited by the system when the ratio $r = \frac{\tau}{T}$ is constant, although the response is temporally scaled by T. Thus if the node had $T = 1.0$ and $\tau = 0.1$, the response would be identical to a node with $T = 10.0$ and $\tau = 1.0$, just 10 times slower. This holds so long as the integration step is sufficiently small to accurately model the dynamics and discretisation of the synaptic delay. Therefore, for the purposes of investigating a single node, we may take the node time constant T as unity, on the understanding that the behaviour may be temporally scaled, but it is qualitatively identical as long as the ratio $r = \frac{\tau}{T}$ is constant. This assumption holds for all of the following analysis in this Chapter. The equation then becomes:

$$\dot{x}(t) = -x(t) + w\sigma(x(t-\tau) + \theta) + I(t) \tag{II.2.7}$$

A number of works (e.g. [11]) have analysed such a system without the inclusion of the time-delays and much of the behaviour observed is the same as the time-derivatives vanish as $t \to \infty$. They provide a starting point for this analysis, but the effects of including delays into the system can significantly alter its behaviour.

The fixed points (x^*) of a standard CTRNN node given by Equation II.1.11 are found when $\dot{x}(t) = 0$, so we can evaluate the expression shown in Equation II.2.8 with a range of self weight and threshold values; the results are

shown in Figure II.2.3 a-e. Figure II.2.3 f shows the results of evaluating Equation II.2.8 when $I = 0$ for a range of weights. This shall become useful in validating the stability behaviour of the CDRNN node considered later.

$$I = x^* - w\sigma(x^* + \theta) \tag{II.2.8}$$

This will hold for non-oscillatory CDRNN systems which in the limit behave as per their CTRNN counterparts. However, as we will address later, it is possible for a single CDRNN node to behave in an oscillatory or even a chaotic manner which may violate this analysis. For this analysis to hold, as $t \to \infty$ then $x(t) = x(t - \tau)$.

A single node undergoing a normal decay to a fixed point in the absence of an external input will approach the weight of the self connection. In this case, without an external input, there remain only three variables which define the behaviour of the node: the weight, threshold and time delay, shown in Equation II.2.9.

$$\dot{x}(t) = -x(t) + w\sigma(x(t - \tau) + \theta) \tag{II.2.9}$$

Figure II.2.4 shows the results of an investigation into the behaviour of a single node under these conditions, for the three different threshold values used in Figure II.2.3. For each configuration of weight and time delay parameter Equation II.2.9 was evaluated for 20000 steps with an integration timestep of 0.01s with an initial condition of 10.0. This was long enough for the ultimate state of the system to evolve (to reach a steady state or to explore the full range of values of the oscillation). The delay value was small enough to smoothly represent the delays. As demonstrated in this Chapter, the evolution of the system dynamics is independent of the initial condition so long as it is not the fixed point of the system. It is worth noting that the period evaluated was sufficient to ensure an independence to the initial conditions. The maximum and minimum values of the output over the next

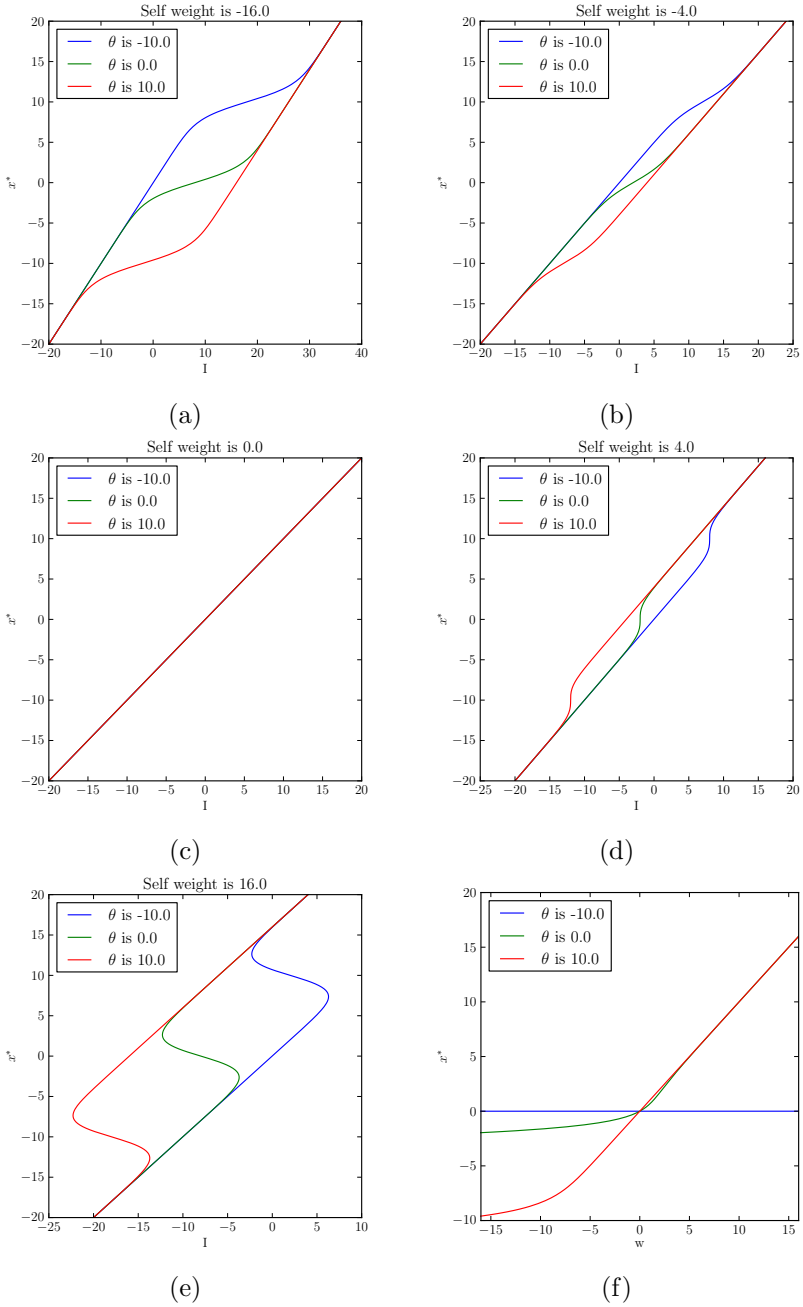


Figure II.2.3: Global stability for a single CTRNN node

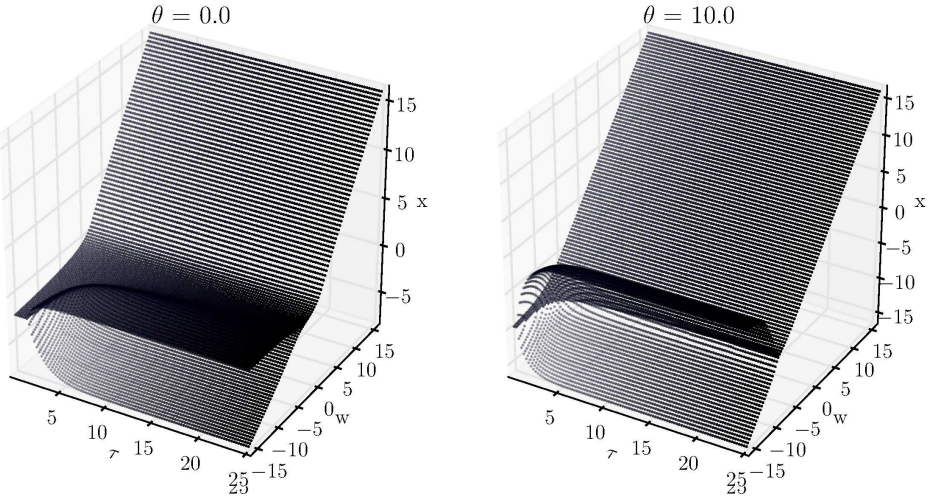


Figure II.2.4: CDRNN Stability

two thousand timesteps are plotted. Any separation of the two surfaces denotes those conditions which lead to long term oscillatory behaviour. The stability plot for threshold of -10.0 is not included because it simply follows the curve shown in Figure II.2.3f. This is because the threshold so biases the output of the sigmoid function that $w\sigma(x(t-\tau)+\theta) \to 0$ as $\theta \to -\infty$, causing the output to decay to zero. The opposite is true when $\theta \to \infty$, as $w\sigma(x(t-\tau)+\theta) \to w$ and the output will therefore approach the value of the self weight. When $\tau = 0$ in Figure II.2.4 a-b, the curves in (w, x^*) follow the stability curves shown in Figure II.2.3f. Where these plots differ from the stability analysis of the CTRNN is in the oscillations which occur when the weight of the self connection is strongly negative and there is a positive time delay. The onset of oscillation is earlier as the threshold increases. As τ increases and w decreases, beyond the threshold values indicating the onset of oscillation, the amplitude of the oscillations increases to a maximum. The oscillations are limited between zero and the (negative) value of the self weight.

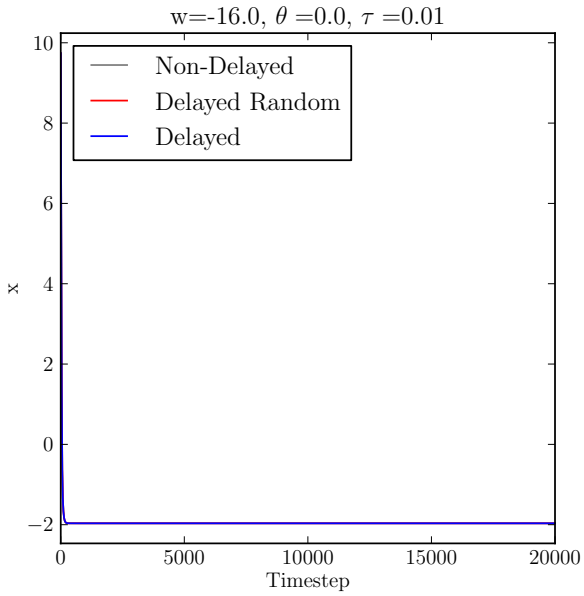
Output trajectories for a self weight of -16.0 and a zero threshold elucidate the evolution of oscillatory behaviour with increasing delay in the

self-recurrent connection, shown in Figure II.2.5. Where the 'Non-Delayed' trace shows the performance of the equivalent CTRNN, 'Delayed Random' shows the behaviour where the initial conditions for the delay were continuously random, and 'Delayed' shows the profile when the delay initial conditions are constant.

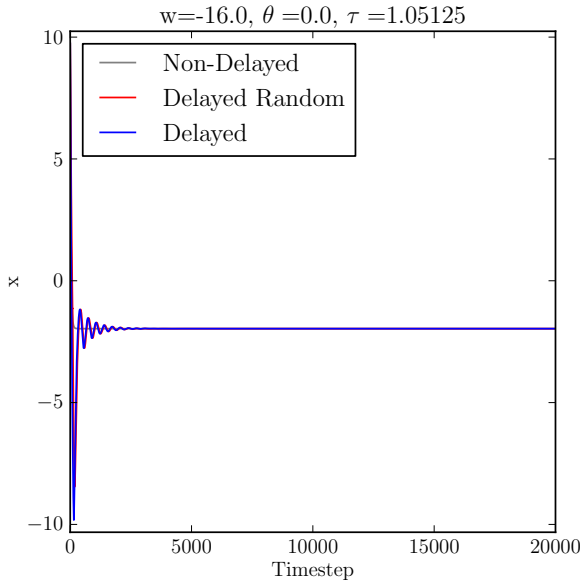
In Figure II.2.5a the output rapidly converges to a stable point, whereas in Figure II.2.5b the output undergoes a highly damped oscillation before converging on the same value. The output of the system is dependent upon both the current value and a past value of the output. In this case, the lag in the output means that the system is injected with a positive input for longer, causing the self connection term to be much larger than the non-delayed system. The strong negative weight causes a rapid drop in the output and by the time that this drop propagates through the delayed connection, the output has overshot the stable point. The system responds to this overshoot in the same way leading to a periodic oscillation. Whilst the initial conditions are still injected into the system (whilst the simulation time is less than the delay), the output will approach the value of the self weight. In Figure II.2.5b the connection delay is less than the rise time to this value and so damps the output. As the time delay increases (See Figure II.2.5 c-d) an un-damped periodic oscillation is set up. Whilst in Figure II.2.5c this appears quasi-sinusoidal, Figure II.2.5 d clearly demonstrates the approach to a stable value for the duration of the propagation delay, imitating a quasi-square wave form. The peak values between which the fully evolved oscillation (the time delay allows a stable value to be reached) varies may be determined by numerically inspecting the governing equations (for this analysis the threshold is taken to be zero). Consider the equation below and an approximation to its numerical integration.

$$\dot{x}(t) = -x(t) + w\sigma(x(t-\tau) + \theta) \tag{II.2.10}$$

$$x(t + \Delta t) \approx x(t) + \Delta t\, \dot{x}(t) \tag{II.2.11}$$

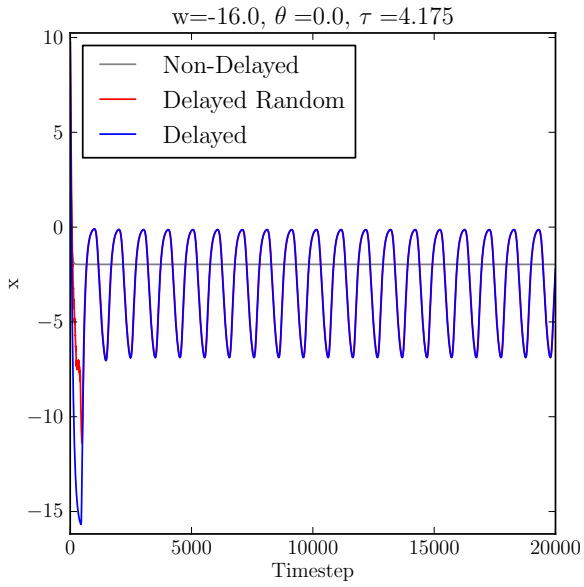


(a)

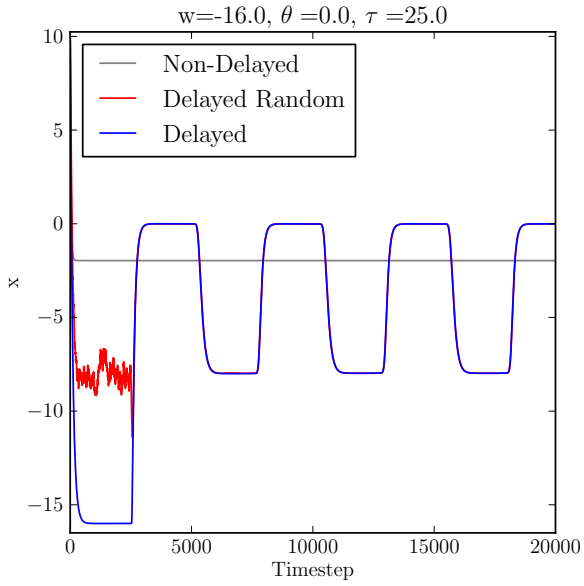


(b)

Figure II.2.5: Profiles of varying time delay



(c)



(d)

Figure II.2.5: Profiles of varying time delay (continued)

If the initial condition, $x(0)$, is positive enough to saturate the sigmoid, then:

$$x(t+\Delta t) \approx x(0) + \Delta t(-x(0) + w\sigma(-x(0))) \quad \text{(II.2.12)}$$

$$\text{as} \quad \sigma(-x(0)) \approx 1 \quad \text{(II.2.13)}$$

$$x \rightarrow w \quad \text{(II.2.14)}$$

If the initial condition $x(0)$ or the output value, $x(t)$, is negative enough to saturate the sigmoid, then:

$$x(t+\Delta t) \approx x(t) + \Delta t(-x(t) + w\sigma(-x(t))) \quad \text{(II.2.15)}$$

$$\text{as} \quad \sigma(x(t)) \approx 0 \quad \text{(II.2.16)}$$

$$x \rightarrow 0 \quad \text{(II.2.17)}$$

But when the output is zero:

$$x(t+\Delta t) \approx \Delta t(w\sigma(0)) \quad \text{(II.2.18)}$$

$$\text{as} \quad \sigma(0) = \frac{1}{2} \quad \text{(II.2.19)}$$

$$x \rightarrow \frac{w}{2} \quad \text{(II.2.20)}$$

Whereupon the output will then tend back to zero and the cyclic process will repeat itself indefinitely. Should the initial conditions be zero, the dynamics simply skip to the last step, as it were. If the initial condition is close to the fixed point of the network the oscillations may take some time to establish but eventually will. The only initial condition which would result in no oscillation is the fixed point itself, as the system remains stable and has no dynamic response. Excluding this case, as long as the weight is sufficiently negative and the delay is long enough the system will always oscillate between zero and half the weight value. This long term behaviour of the system is very robust to changing initial conditions and previous delay states and initial deviations are damped out over the first oscillation or so (as shown in Figure II.2.5). However, this is something of a special case. With the inclusion of the threshold and input terms into the equation

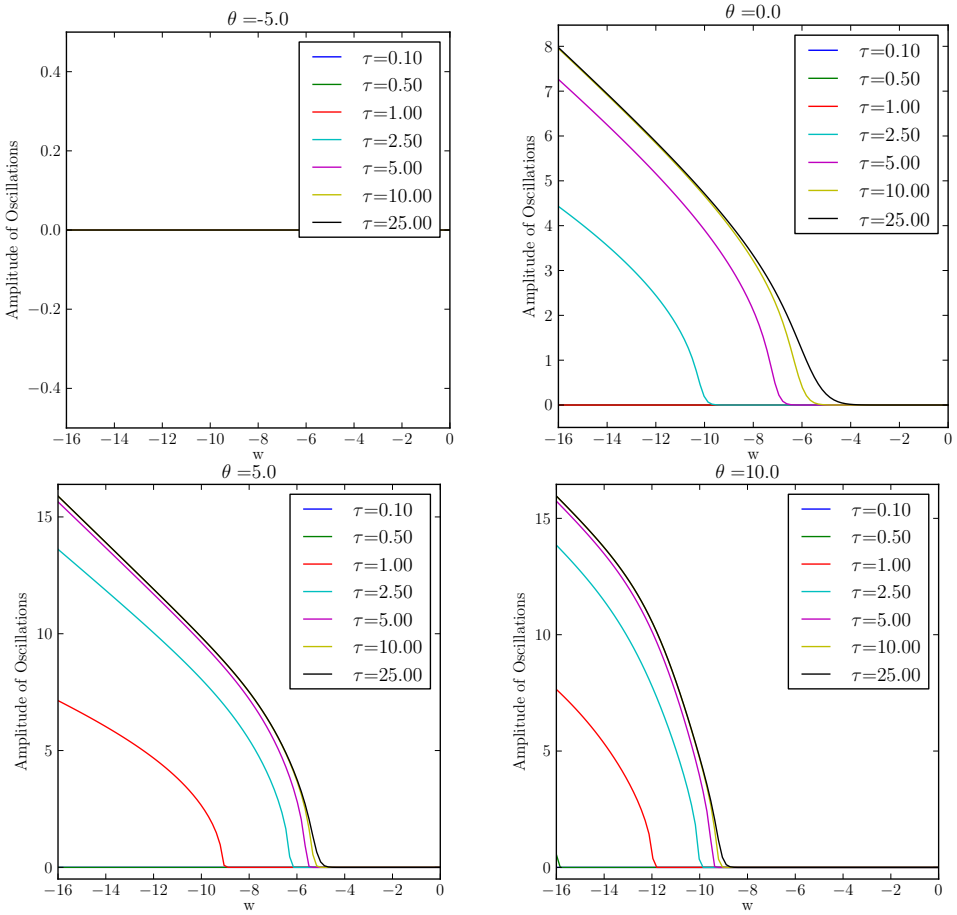


Figure II.2.6: Amplitude of Oscillations

the behaviour changes, as is discussed in detail later. If the threshold is positive, it biases the response of the system when the output is zero, as the bias will tend to saturate the sigmoid and cause the output to tend to the weight value, as opposed to half this. This can be seen in Figure II.2.4b as the oscillations vary between zero and -16.0 at their peak. Figure II.2.6 shows the amplitude of the resulting oscillations for a range of weight, time delays and threshold values. The output of Equation II.2.9 was evaluated for 100,000 time steps and the maximum and minimum values of the output recorded for the last 50,000 timesteps and the amplitude calculated as the difference between them.

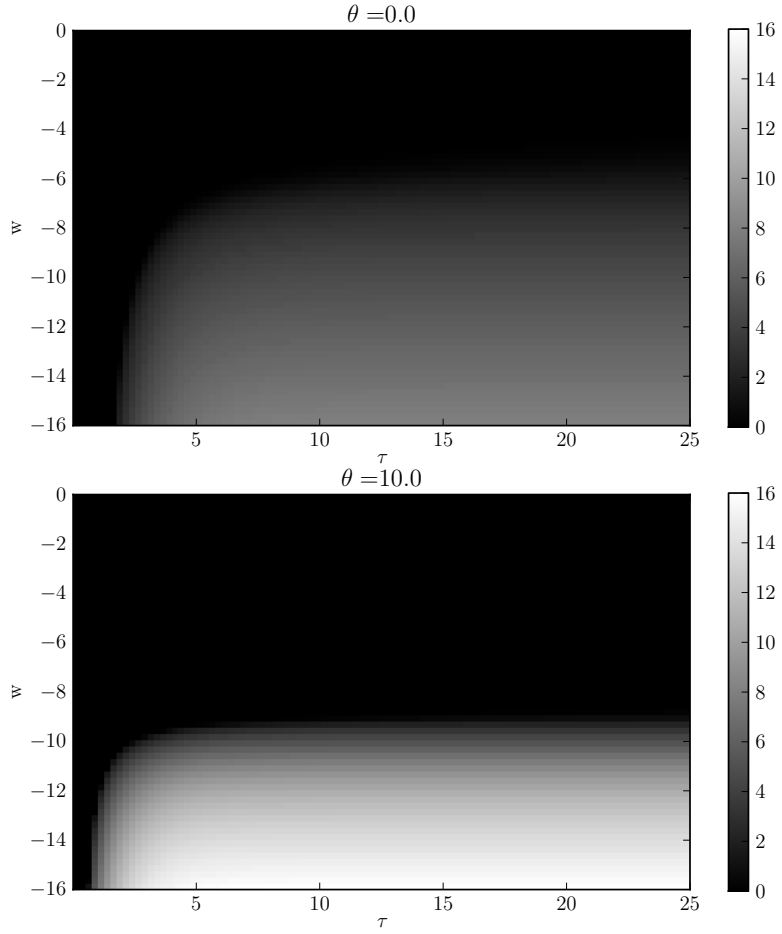


Figure II.2.7: Oscillation Zones for Different Thresholds

Looking at this differently, we can plot the amplitude of the oscillations against weight and delay to determine the boundary of the oscillatory zone, shown in Figure II.2.7. Figure II.2.8 shows the global stability analysis of Figure II.2.3, but for a CDRNN node with a range of time delays for a threshold (θ) of zero. As with the previous analysis the maximum and minimum points of the output trajectory for a period after it has settled down into its long term behaviour are recorded and plotted. Where the maximum and minimum values are not the same (i.e. an oscillation is present) the data is coloured differently. For non-oscillatory modes this is the long way around to finding x^* where $\dot{x^*} = 0$ rather than directly from the equation.

However, this method allows the visualisation of oscillatory modes directly with a constant input to the node. Each sub-figure illustrates the results for a different self-weight. The zero delay lines exactly reproduce the zero bias curves shown in Figure II.2.3. When the self weight is zero the fixed point x^* is simply the input to the node and is therefore omitted here. Specifically Figure II.2.8d reproduces the fold in the stability curve. The unstable portion of this curve cannot be visualised by this technique as only the long term stable points are recorded, but by repeating the simulations with initial conditions of both +16.0 and -16.0 we may illustrate the dynamics from both ends of the curve, with some points showing the vertical drop (or rise) to the next stable surface. The folds in the equilibrium surface form the bifurcation set of Equation II.2.7.

The derivation of the cusp bifurcation set hold for a CDRNN as per the CTRNN analysis undertaken by Beer [10]. The cusp is defined by the simultaneous zeros of $f(y, w, \theta, I)$ and $\dot{f}(y, w, \theta)$, and can be solved to find;

$$I = \pm 2\text{sech}^{-1}\left(\frac{2}{\sqrt{w}}\right) - w\sigma\left(\pm 2\text{sech}^{-1}\left(\frac{2}{\sqrt{w}}\right)\right) - \theta \qquad \text{(II.2.21)}$$

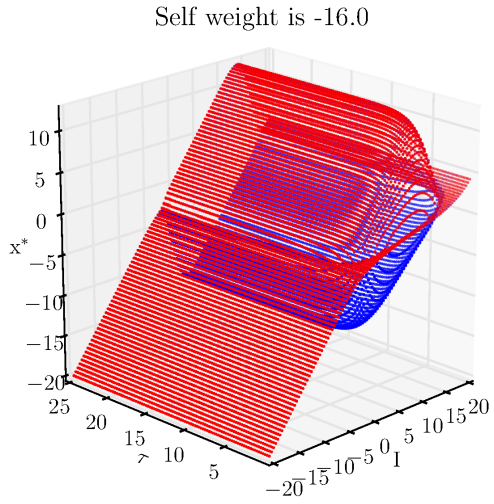
This can be simplified for $w \geq 4$ to;

$$\text{lb}(w, \theta) \equiv 2\ln\left(\frac{\sqrt{w} + \sqrt{w-4}}{2}\right) - \frac{w + \sqrt{w(w-4)}}{2} - \theta \quad \text{(II.2.22)}$$

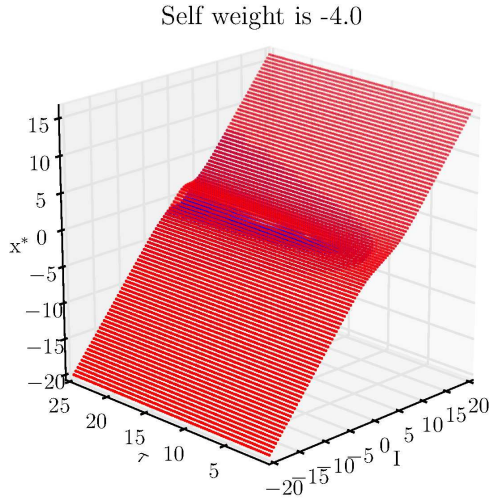
$$\text{rb}(w, \theta) \equiv -2\ln\left(\frac{\sqrt{w} + \sqrt{w-4}}{2}\right) - \frac{w - \sqrt{w(w-4)}}{2} - \theta \quad \text{(II.2.23)}$$

The dynamics of a single CDRNN node with constant input and strong negative self-weight, shown in Figure II.2.8d, goes through three distinct phases shown in Equation II.2.24, the boundaries of which are described by Beer [10].

$$(x^* \to I) \quad \overleftarrow{I \ll -\left(\frac{w}{2} + \theta\right)} \quad \left\{ \begin{array}{l} x^+ \to I \\ x^- \to I + w \end{array} \right\} \quad \overrightarrow{I \gg -\left(\frac{w}{2} + \theta\right)} \quad (x^* \to I + w) \qquad \text{(II.2.24)}$$

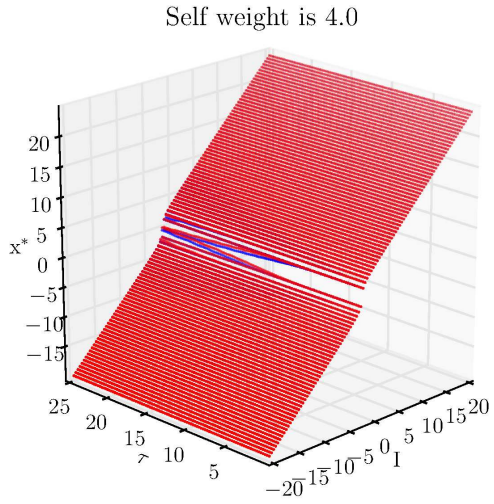


(a)

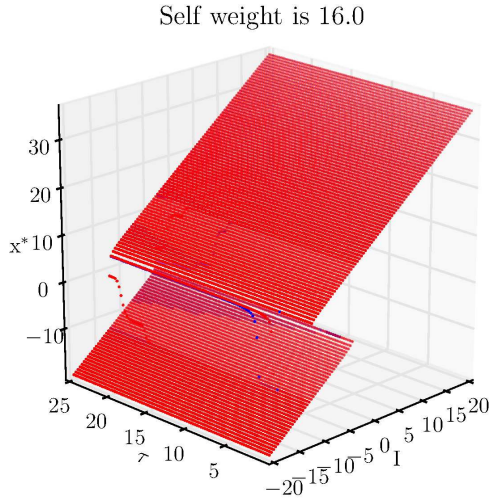


(b)

Figure II.2.8: Input stability for a single CDRNN node



(c)



(d)

Figure II.2.8: Input stability for a single CDRNN node (continued)

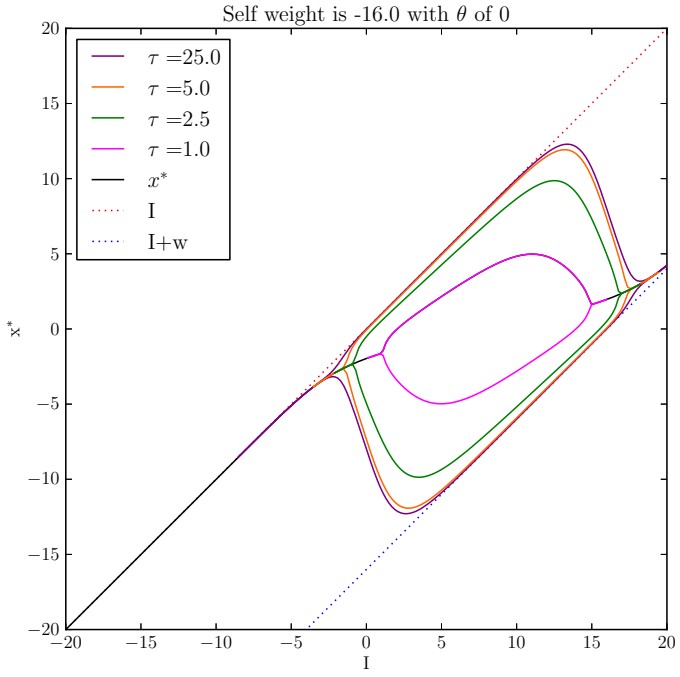
Where x^* is the stable fixed point of the system, and x^+ and x^- are the positive and negative extents of the oscillations.

This can be seen most clearly graphically by taking sections through Figure II.2.8 for values of constant delay, as shown in Figure II.2.9 for a range of self-weights. Figure II.2.9a shows the bounded evolution of oscillation, where the limits are as described in Equation II.2.24, illustrated by the plotting of lines of I and $I + w$. As the weight decreases the region of oscillation diminishes and reduces in amplitude as shown in Figure II.2.9b. Along with the reducing size of the oscillatory zone, the delay required for the oscillations to reach the boundary values of I and $I + w$ increases, until even very long delays fail to achieve this (Figure II.2.9c). By the time that the self-weight increases to -2.0, no discernible oscillation is present for the delay values evaluated. Note however that some minute oscillation was seen, illustrated by the change in colour of the lines through what is usually the oscillatory region. If there were no difference, the fixed point line x^* would continue throughout as shown in Figure II.2.3. As with Figure II.2.3, the effect of altering the threshold value simply shifts the curves an equal amount along the input scale, i.e. the same behaviour is seen with ($I = 0$, $\theta = 0$) as at ($I = -10$, $\theta = 10$).

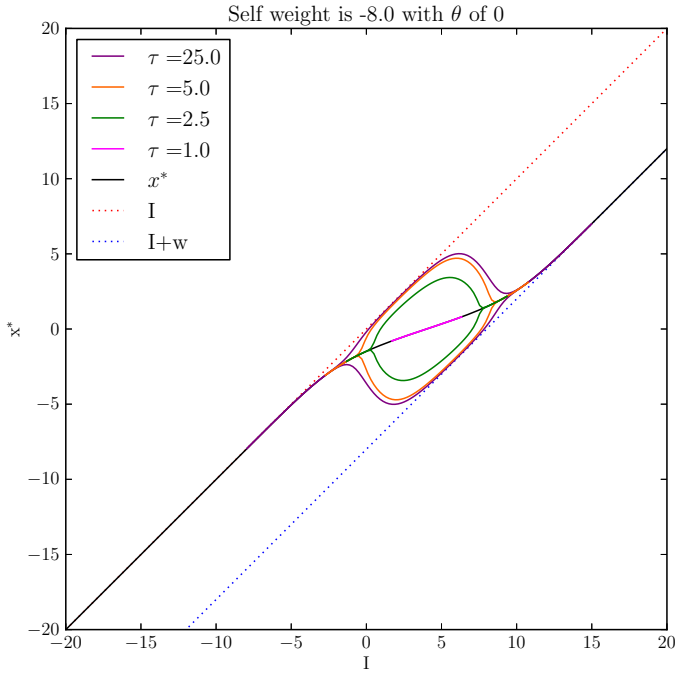
The boundaries of the oscillatory region in I are observed at the intersection of the three definite regions of a CTRNN node stability surface. Either side of the effect of the sigmoid the stability function $\bar{x}(I, w, \theta)$ is I and $I + w$. In the middle an approximate expression can be derived for when $x + \theta \approx 0$, at the point $\hat{I} = -\left(\frac{w}{2} + \theta\right)$. If this is the case, then $\sigma(x + \theta)$ can be replaced by its Taylor expansion [10].

$$\sigma(x + \theta) \approx \frac{x + \theta}{4} + \frac{1}{2} \tag{II.2.25}$$

Which can be substituted into Equation II.2.8 to obtain an approximation

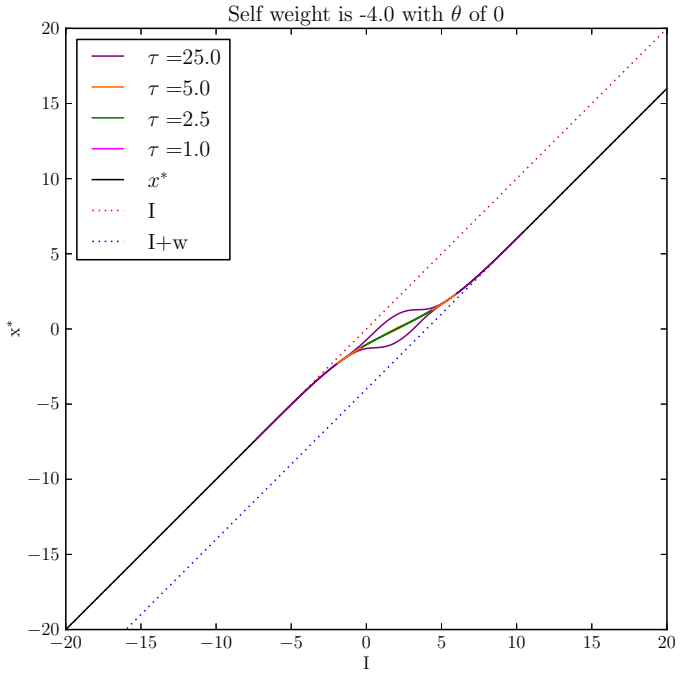


(a)

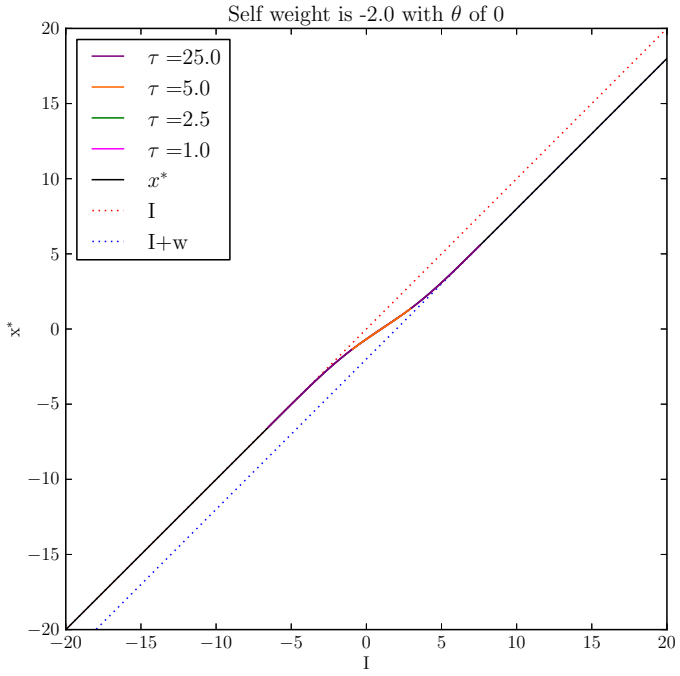


(b)

Figure II.2.9: Input stability cross-sections



(c)



(d)

Figure II.2.9: Input stability cross-sections (continued)

for a CTRNN node.

$$\bar{x}(I, w, \theta) \approx \frac{4}{4 - w}\left(I - \hat{I}\right) - \theta \tag{II.2.26}$$

The intersection of these three straight lines are at $I = -\theta - 2$ and $I = -\theta - w + 2$.

We have described the limit values of the oscillation, between I and $I+w$, but this is the limit of the behaviour. As can be seen in Figure II.2.9 as τ and w reduce from the values required to reach the boundaries, the size of the zone reduces both in I and x^*. There are two effects superimposing to generate the complex 3D surfaces seen in Figure II.2.8.

Consider the behaviour of a single node which has been at zero for a period greater than the delay τ. Assuming that for each oscillation the delay is long enough for the output to reach a stable value and all the while the delayed input is injecting the previous stable state into the node, we can write a non-linear recursive sequence for the bounds of the oscillation.

$$a_n = w\sigma(a_{n-1} + \theta) + I \tag{II.2.27}$$

Where a_n is the stable value reached for oscillation $n \in \mathbb{Z}$.

This non-linear recursive equation is impossible to solve generally. However, for oscillations that take place within the central region of the stability curve we can approximate the sigmoid function by terms in its Taylor series. Thus we can rewrite Equation II.2.27 as follows:

$$a_n \approx w\left(\frac{a_{n-1} + \theta}{4} + \frac{1}{2}\right) + I \tag{II.2.28}$$

Ignoring the input term, it is then possible to arrive at a particular closed form solution of the above:

$$a_n \approx \frac{(\theta + 2)4^{-n}w(w^n - 4^n)}{w - 4} \tag{II.2.29}$$

Where $n \in \mathbb{Z}$ and $|w| < 4$. Figure II.2.10 shows the results of evaluating Equation II.2.27 for the first 50 oscillations for a range of weights with

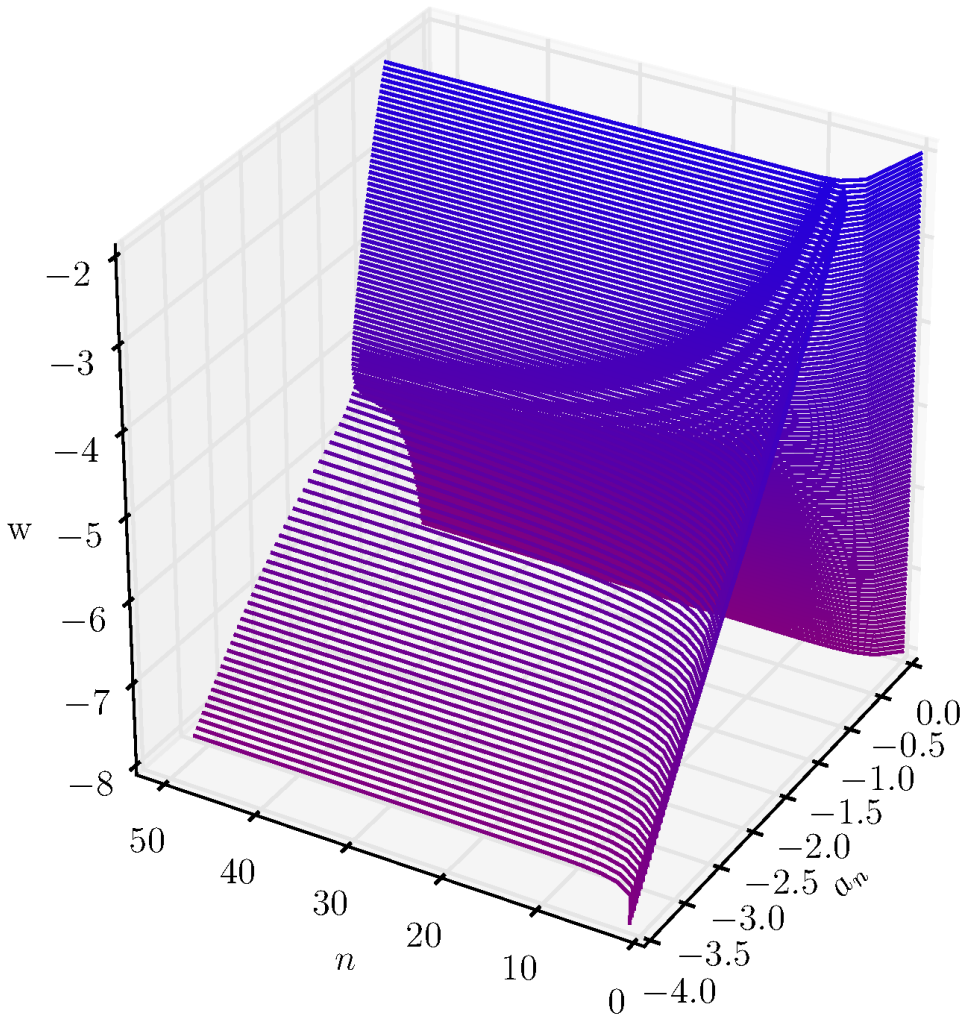


Figure II.2.10: Recursive Oscillation Surface

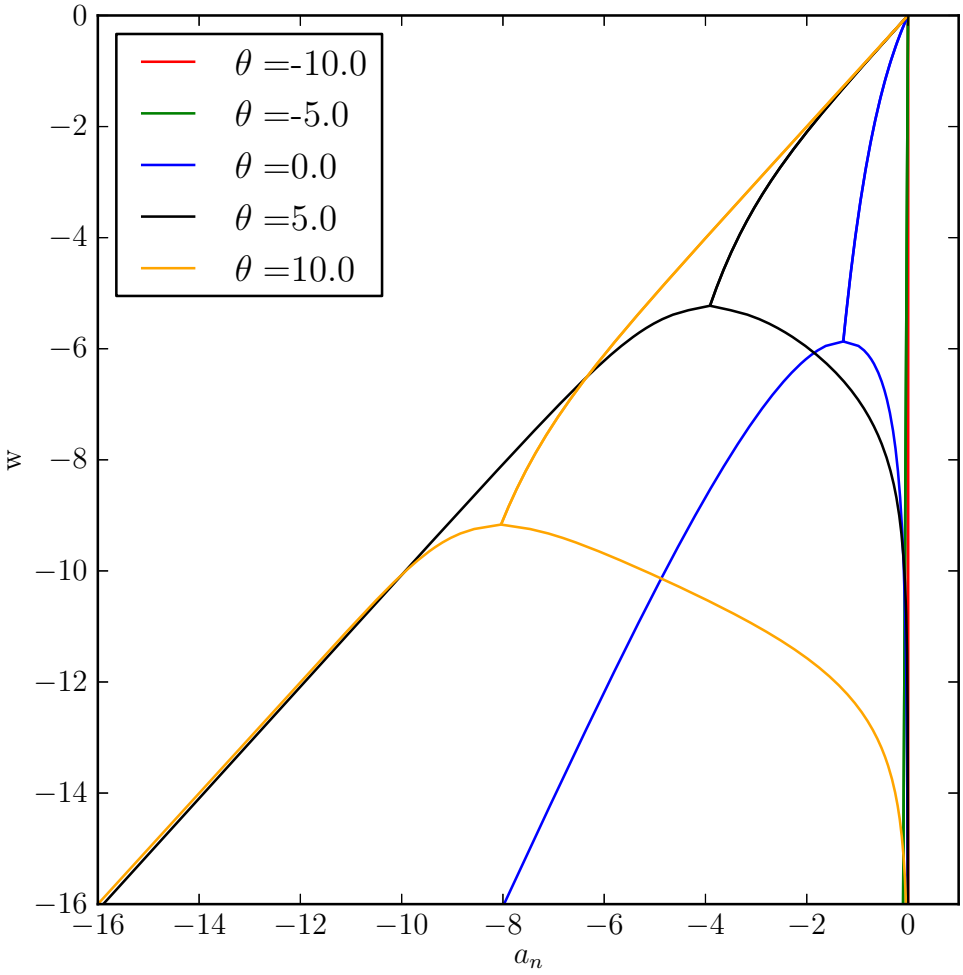


Figure II.2.11: Bifurcation Plot

zero threshold and input. Divergence of the two surfaces demonstrates the presence of oscillation. When a stable oscillation is present the bounded dynamics asymptotically approach a value symmetrically offset from the equivalent fixed point of an identical CTRNN. These curves exactly correspond to the damping witnessed in Figure II.2.5 and are bounded between I and $I+W$ as before. As the weight increases, the oscillations reduce until the dynamics are stable at the fixed point, as in Figure II.2.3. If we consider the limits of the behaviour as $n \to \infty$, a bifurcation is observed (see Figure II.2.11) at a critical value of weight which marks the transition from stable to

oscillatory behaviour. Whilst it appears that the negative threshold curves shown never reach the bifurcation point, they do indeed but at much lower values of self weight (around -400 and -60,000 for $\theta = -5.0$ and $\theta = -10.0$ respectively). As in ER we are typically interested in weights within the bounds $-16 \leq w \leq 16$ [10] they are not applicable but they follow the same qualitative behaviour. Without a delay value long enough this change in behaviour of the system cannot happen.

Secondly, from one periodic oscillation to another the recurrent input through the self connection is invariant for the duration of the time delay. The system then behaves as per a simple first order response only complicated by the action of the sigmoid function which non-linearly affects the output. If the time delay is not sufficient for the output to reach the next stable value (as previously discussed) before the delayed injection of the time variant response just undertaken by the output, then the amplitude of the oscillation is reduced. If the delay is too short, then the oscillation will be damped out completely as seen in Figure II.2.5b.

The combination of these two mechanisms fully defines the region of oscillation and the asymptotic values of any particular solution of the recursive equation define the long term values of the oscillations. In this way the full range of dynamic behaviour for a single CDRNN node has been characterised and explained. Through the simple inclusion of a continuous delay on a self-recurrent connection a single node may generate long term oscillatory behaviour in the manner of a pattern generator. A central hypothesis to this Thesis is that in this way the CTRNN model can be simply extended to generate complex behaviour and further dynamic complexity which may be exploited by evolution of which the traditional model is not capable [15]. Indeed, it is encouraging that oscillations have been observed in even very small cultured biological neural networks [97], supporting the bio-inspired approach adopted throughout this Thesis.

II.2.3 Beyond a Single Node

The effect of combining nodes to form a larger network on the dynamics is highly complex, and difficult to consider analytically. Certainly beyond a three degree of freedom system visualising analysis becomes difficult. Beer considers in detail the dynamics of larger systems with respect to bifurcations and stability [10]. For the most part this analysis and decomposition approach to the consideration of larger systems holds true as the bifurcation for positive weights ($w > 4$) is not affected by the oscillations. Without considering this subject in too great a detail, there are some general conclusions we can draw as to the difference between Beer's analysis and that which would hold true for the delay systems considered here. Beer discusses hysteresis in non-autonomous circuits where the neuron is driven across the folds in the saddle-node bifurcation. For a delayed system this could occur when an autonomous oscillatory node is taken as the input and so can be achieved without a time-variant input to the system. Secondly, if an autonomous oscillatory node (in effect a pattern generator) takes input from another node in the system, this input can suppress or express the pattern depending on its value, an illustrative example is shown in Figure II.2.12. This happens in the same manner as an external input would, where in the case of the single node system $I = w\sigma(x(t-\tau))$. This would enable a network to switch pattern expression on and off at will which one could imagine being highly useful in the emergence of complex adaptive behaviour.

II.2.4 Discretising a Delayed Continuous System

The non-linear systems in which we are interested are continuous, but any study which includes time variant inputs must be numerical. There are a wide array of numerical methods for integrating differential equations, but when the dynamic systems modelling neural networks are embodied in

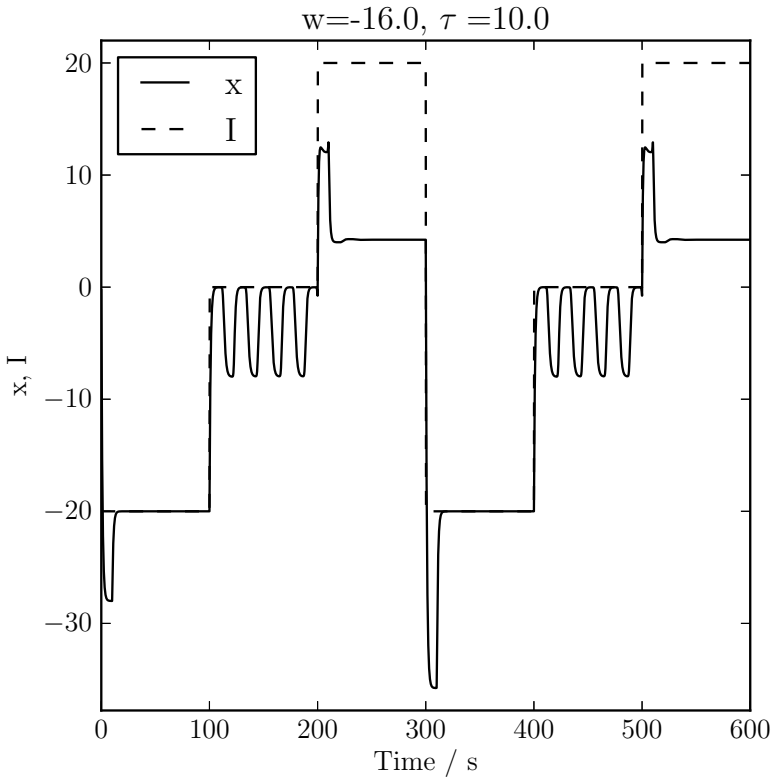


Figure II.2.12: Switched Oscillations

computer simulations our choice becomes very limited. Appendix V.A introduces two simple methods starting with the most basic of all, the Newton-Euler method, which has simplicity as its main virtue. However it is used exclusively in the majority of literature in ER, particularly that on CTRNNs. Whilst the error associated with this first order method, $\mathcal{O}(h)$, is larger than that of more sophisticated methods (the order of accuracy for a fourth-order Runge-Kutta method is $\mathcal{O}(h^4)$), maintaining a small step size $h \ll 1$ for accurate physical simulations and delay modelling reduces the significance of this. To capture small variations in time delay and accurately approximate the continuous system a smaller time step than is usual for ER must be employed, although an optimal choice of time step for studying DDEs is often between 0.001s and 0.05s [70].

Specifically we are interested in numerically evaluating Equation II.2.3 and governing its parameters using evolutionary methods. Beyond the numerical integration of the system of first order non-linear differential equations, we must be able to model the continuously delayed outputs governed by the synapse delay times τ. The investigations above evaluate only delay values which are exact multiples of the integration step, but for evolved systems this is not the case. The outputs of the continuous system are approximated at intervals of Δt. For a time delay of τ seconds, we must store the values of each output for $\tau/\Delta t$ time steps. However, we must consider that $\tau \in \mathcal{R}$ and so can take values that are not multiples of Δt. As we can only store output values at each time step and there is a limited amount of memory to record previous output states, an approach was adopted where each synapse delay was represented as a First-In-First-Out (FIFO) store. Outputs at each time step are appended to each list and the first entry removed to maintain a store of constant length. The store is initially created with a length of $\tau/\Delta t$ rounded up to the nearest integer and populated with the initial value for the node outputs (usually zero). When the delayed out-

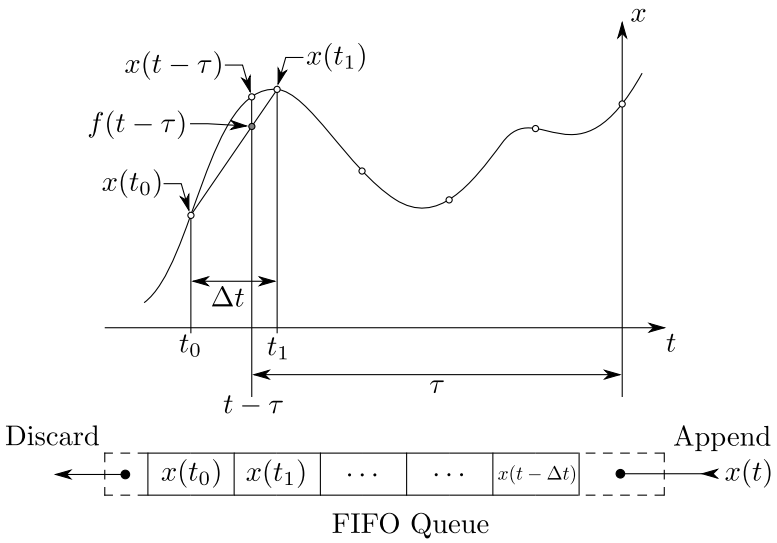


Figure II.2.13: Discretely Modelling a Continuous Delay

put of a synapse is sought the value is linearly interpolated between proximal outputs in the queue, as shown in Figure II.2.13.

The estimate of the nodes output at $t - \tau$ is therefore:

$$f(t-\tau) = x(t_0) + (x(t_1) - x(t_0))\frac{\tau(\text{mod } \Delta t)}{\Delta t} \tag{II.2.30}$$

The error between the true continuous output and the linear interpolation, $R_T = f(t) - x(t)$, is bounded as shown in Equation II.2.31.

$$|R_T| \leq \frac{(t_1 - t_0)^2}{8} \max_{t_0 \leq t \leq t_1} |f''(t)| \tag{II.2.31}$$

It is eminently possible to adopt a higher order polynomial interpolation in an effort to reduce the error. However, it has been found that this approach produces a minimal improvement at best, and can lead to unrealistic spikes in the output. The crucial parameter in determining the accuracy of the numerical approximation is the effect of the time step size on the Newton-Euler integration. When considering this, alongside the fairly significant increase in computational effort required, the simple linear interpolation method has been adopted throughout this research.

II.2.5 Conclusions

In this Chapter an argument has been set forth for increasing the complexity of continuous dynamic neuron models to incorporate synaptic time delays. This is because they are thought to be important in contributing to the dynamics of natural neural systems and they are not included in the neuron models used in ER. Such dynamic systems are classified as a subset of Delay Differential Equations, which are effectively infinite dimensional systems even with only a single delay. A new extension to the standard CTRNN neuron model has been presented incorporating a continuous delay on each incoming synapse.

Through dynamic analysis the dynamic differences between comparable networks have been illustrated with and without delays, emphasising the potential increase in complex dynamic behaviour which may be exploited by artificial evolution for adaptive behaviour. Other studies of similar delayed dynamic equations have been reviewed and a stability analysis for a single neuron system has been undertaken.

The full range of dynamics of which such systems are capable have been comprehensively evaluated and explained, with particular reference to the oscillatory modes present in the gamut of possible behaviours. The boundaries of the oscillatory zone have been numerically explored, the conditions required to bring this about commented upon with how this understanding can be used to guide our search for interesting solutions in ER.

Chapter II.3

Minimalistic Simulation of a Quadruped Robot

Studies in ER often couple the investigations of adaptive behaviour and locomotory gaits for legged robots [15], and they are two of the most common applications for continuous recurrent networks. Early research tended to focus on simple hexapod insect-like architectures which are highly stable platforms, with later developments considering more dynamic structures. Chapter II.2 explored the dynamics of small systems of DDEs and highlighted the enhanced oscillatory behaviours present over the commonly used CTRNN model. It is natural therefore to suppose that these delayed networks may provide a benefit in the development of evolved gaits, and to explore their application in legged robotics. The aim of this Chapter is to develop the simulation tools to be able to evaluate later experiments where delayed networks are used to control the gait of a quadruped robot in Section III.3.2.

The design, simulation and analysis of legged robots for the development and analysis of gaits is well established. The tools for this tend to be either One-Dimensional (1D) simple approximations or fully featured dynamic simulation environments which carry a high computational overhead [52]. The

first are by nature somewhat limited and whilst providing useful tools for appreciating dynamic gaits do not address the crucial issues of 3D stability. Full simulation environments, of which there are many, provide a fully featured 3D dynamic analysis but likely have a degree of extraneous overhead inflating their computational cost. Additionally, integration of these third party software packages into home-grown code for research or development places significant limitations on the researcher.

Evolutionary methods are characterised by the need for a large number of trials over successive generations as an optimal solution is sought. Our requirements of any simulation must therefore not only be for a realistic portrayal, but also for computational efficiency. This central tenet of ER was first formalised by the work of Jakobi [59, 60] in his work with two-dimensional models of wheel robot locomotion.

There are a number of mature simulation packages used in such research. However, the inclusion of delayed and pattern networks (see Chapters II.5 and II.6) require a highly tailored evolutionary architecture which has been developed in Python. To accelerate this code for the investigation of long evolutionary runs, the packages are translated using PyPy into C-executables. This provides a massive speed-up, but does limit the ability to integrate external software. Hence the need to develop a simple system which can accurately and efficiently evaluate the performance of evolved gait controllers. This prevented the use of traditional simulation packages and lead to the development of the minimalistic dynamic simulation presented herein.

We should however be absolutely clear as to the role of the simulation developed. There is little novelty in the mathematical formulation of the system. Indeed this is quite intentional as the focus from the start has been on simplicity. What is novel is the minimalistic treatment in the development of the simulation, in line with the approach of Jakobi, to generate a computationally efficient way of evaluating evolved gaits. There is however a

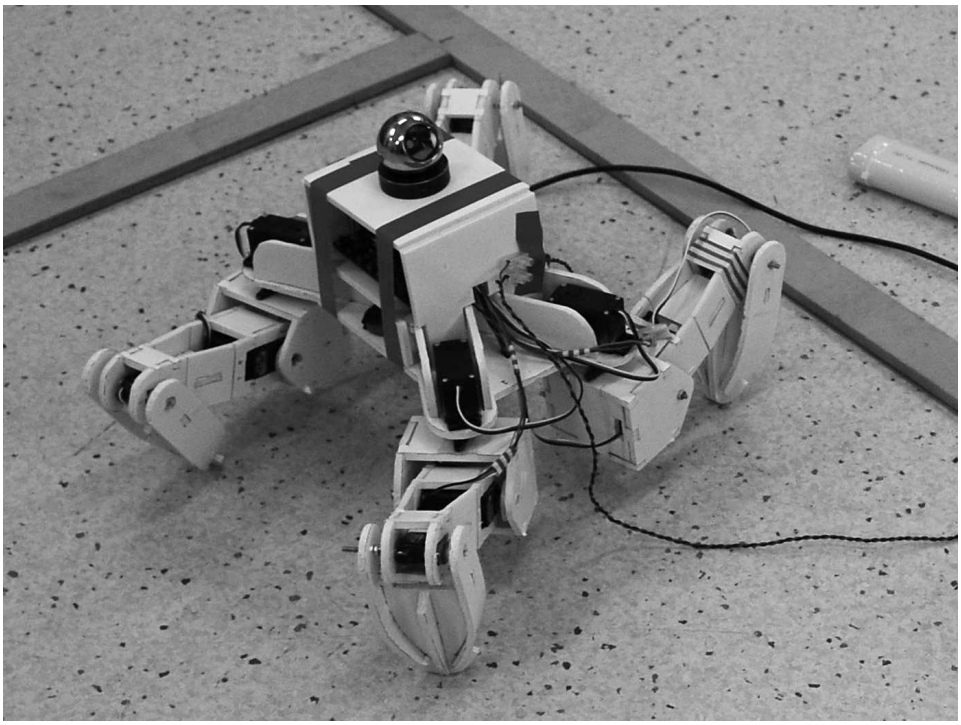


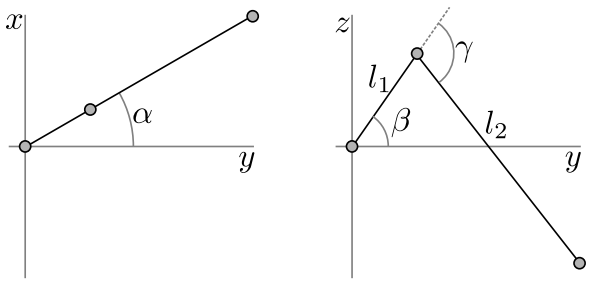
Figure II.3.1: Real Robot for Verification of Simulation Results

danger that oversimplification could lead to unstable or unrealistic dynamics that cannot reliably simulate the quadruped gait.

II.3.1 Single Leg System

Initially, we shall consider the geometry and control of a single leg which may later be used in a modular manner to model a whole range of robotic organisms (such as that shown in Figure II.3.1 which was developed to verify the simulation developed herein). We shall model a sprawled leg geometry, common to all manner of insects and reptiles, which is the most primitive of postures. The alteration of these expressions to represent a more traditional upright stance is simple and may be used to easily simulate a range of mammalian forms.

The leg shown in Figure II.3.2 consists of two rigid bodies fixed in the



(a) Top View (b) Side View

Figure II.3.2: Diagram of a Single Leg

main body reference frame by a spherical joint constrained to rotate around the z-axis and the x-axis only. A revolute join between the two limbs allows relative rotation only in the local x-axis for the bodies. This approximates a mechanically realisable form of tetrapod limb [91]. Realistic limits of $-\frac{\pi}{4} \leq \alpha \leq \frac{\pi}{4}$, $-\frac{\pi}{2} \leq \beta \leq \frac{\pi}{2}$ and $0 \leq \gamma \leq \pi$ are set on the range of motion of the leg. In this coordinate system, relative to the hip joint, the coordinates of the foot are (assuming absolute limb movement):

$$x = \sin(\alpha)\left(l_1 \cos(\beta) + l_2 \cos(\gamma - \beta)\right) \quad \text{(II.3.1)}$$

$$y = \cos(\alpha)\left(l_1 \cos(\beta) + l_2 \cos(\gamma - \beta)\right) \quad \text{(II.3.2)}$$

$$z = l_1 \sin(\beta) - l_2 \sin(\gamma - \beta) \quad \text{(II.3.3)}$$

This can be assembled into a system which is a 3D analogue of a 1D peg-leg walker or Chaplygin sled [52], where the leg is attached to a solid body constrained to move along an axis. This is similar to the 'rail roach' approach of Ghigliazza [40, 41] which was a first step towards a hexapod model and used single limb telescopic legs, rather than the three-axis leg considered here.

A natural extension of the single leg system is to move towards a more realistic consideration of quadruped, or hexapod, legged robots. This requires a thorough treatment of the physical dynamics, as we can no longer make the

simplifying assumption of a symmetrical gait and furthermore require that the robot body is not used as a support whilst all legs are simultaneously out of contact with the ground.

Furthermore we have assumed absolute limb movement in the formulation of the above Equations for foot position. Real robot limbs tend to suffer to a greater or lesser extent from backlash, where there is some play in the joints due to the motors or flexibility of the limbs. This means that strictly the above assumption is rendered invalid. However, whilst there are techniques such as compliance methods which can be used to model these, the additional complexity of simulation required itself invalidates the fundamental maxims of our approach to a minimalistic simulation. In our desire to maintain a simple single step solution to the body forces and torques at each time step it is undesirable to include such iterative techniques. Some of the play in joints may be approximated roughly in the design of the ground spring constants and simulation damping discussion in the next Section.

II.3.2 Rigid Body Dynamics

The kinematics and dynamics of rigid body systems have been well studied, and may be simulated using a variety of formulations. Classically Newton-Euler constrained Equations of Motion (EOM) [48] are used or more recently Screw Theory [5, 36] has been used heavily in rigid body / robotic simulation. The Newton-Euler approach to maintaining a hierarchical system of reference frames may be used, or formulated in Denavit-Hartenberg (DH) Parameters for the links of a spatial kinematic chain [87].

Dynamic analysis using the Newton-Euler EOM requires the solution to the following set of second-order non-linear differential equations, presented in Equation II.3.4 in matrix form. (Many of the formalisms presented herein make use of certain vector and matrix identities which are presented for completeness in Appendix V.B. Conventions and expressions for the general

representation and transformation of 3D translations and rotations using Euler Parameters are given in Appendix V.C):

$$\begin{bmatrix} \mathbf{M} & \mathbf{0} & \mathbf{\Phi_r^T} \\ \mathbf{0} & \mathbf{J'} & \mathbf{\Phi_\pi^T} \\ \mathbf{\Phi_r} & \mathbf{\Phi}_{\pi'} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{r}} \\ \dot{\boldsymbol{\omega}}' \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{F^A} \\ \mathbf{n'^A} - \tilde{\boldsymbol{\omega}}'\mathbf{J'}\boldsymbol{\omega}' \\ \boldsymbol{\gamma} \end{bmatrix} \tag{II.3.4}$$

Where $\mathbf{M}$ is the mass matrix, $\mathbf{J'}$ is the inertia matrix in the body frame, Φ_r and $\Phi_{\pi'}$ are constraint matrices, $\mathbf{F^A}$ is the applied force vector, $\mathbf{n'}$ is the applied torque vector in the body frame, ω' is the body rotational speed vector and $\boldsymbol{\gamma}$ is the right hand side of the acceleration equation.

For a still minimalistic treatment of a quadruped robot, as shown in Figure II.3.3, these matrices must be assembled for nine bodies, resulting in a matrix of size 126x126 which must be solved. This system is not trivial to solve and has been the focus of a good deal of dedicated analysis software, such as ADAMS, which has been used frequently in the simulation and analysis of legged locomotion, for example by Ho et al. [50]. The aim here however is to derive a simplified and computationally minimalistic analysis without having to rewrite a full dynamics engine. In this way the model can be written in almost any programming language to suit almost any platform to meet the needs of any researcher. The aim of this is to enable the easy integration and modification of the algorithm to maximise ease of investigation and experimentation, without the need of integrating third-party software. The development of this scheme is presented in as a complete a manner as possible to facilitate its use.

Rather than approach the problem as having nine bodies, we shall make a simplifying assumption that the limbs of the robot are massless. This is clearly not accurate, but the body of the robot must necessarily contain the significant majority of the robot e.g. batteries and on board computation. In insects as in many robots, the masses of the legs are very small (Soygunder and Ali created a robot where the leg masses are around one percent of

the total body mass [103], and in the cockroach *Blaberus discoidalis* the ratio is around 5% [66]). The mass and inertial contributions of the legs are often ignored for simplicity [52]. Also, the robot is assumed to be physically symmetrical around the (x,z) plane and the (y,z) planes when adopting a neutral stance. Whilst different leg extensions move the Centre of Mass (Centre of Mass (COM)) of the leg, the overall impact on the location of the body COM is likely small. The inertial effect of leg movement relative to the body will be small given their low mass and low maximum speed.

Here on, we shall follow the classical Newton-Euler approach for a number of reasons. Assumptions have been in order to simplify the simulation of the leg model in the previous Section and we do not wish to introduce further complexity. As we do not wish to consider the contribution of the legs to the dynamics beyond providing impulses at the point of contact, there is no need to consider the kinematic chain of each limb; simple trigonometry within the body reference frame, as in the previous section, is perfectly adequate. The classical approach therefore reduces the complexity of the mathematical assembly of the system which is a key concern in the drive behind its selection.

By making this assumption we can model the system as a single body without constraints and simulate the leg dynamics and ground reactions by application of external forces and torques to the body. For a single body system which is unconstrained the EOM reduce to a system of six equations only:

$$m\ddot{\mathbf{r}} = \mathbf{F} \tag{II.3.5}$$

$$\mathbf{J}'\dot{\boldsymbol{\omega}}' = \mathbf{n}' - \tilde{\boldsymbol{\omega}}'\mathbf{J}'\boldsymbol{\omega}' \tag{II.3.6}$$

Where $\mathbf{F}$ and $\mathbf{n}$ are the forces and torques applied to the system by the legs of the robot and their contact with the ground.

As ω' cannot be integrated directly, we must make use of the relationship

to the Euler parameters:

$$\dot{\mathbf{p}} = -\frac{1}{2}\mathbf{G}^T\dot{\boldsymbol{\omega}}' \tag{II.3.7}$$

The mass and inertial characteristics of a uniform body are given in Appendix V.D. Using the EOM in the form shown in Equations II.3.5 and II.3.6 we can easily rearrange these expressions to find $\ddot{\mathbf{r}}$ and $\dot{\boldsymbol{\omega}}'$ directly in a form suitable for numerical integration:

$$\ddot{\mathbf{r}} = \frac{1}{m}\mathbf{F} \tag{II.3.8}$$

$$\dot{\boldsymbol{\omega}}' = \mathbf{J}'^{-1}\left(\mathbf{n}' - \tilde{\boldsymbol{\omega}}'\mathbf{J}'\boldsymbol{\omega}'\right) \tag{II.3.9}$$

For completeness we shall write these fully using the inversion of a diagonal matrix as given in (V.D.6):

$$\begin{bmatrix} \ddot{r}_x \\ \ddot{r}_y \\ \ddot{r}_z \end{bmatrix} = \frac{1}{m}\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} \tag{II.3.10}$$

$$\begin{bmatrix} \dot{\omega}'_x \\ \dot{\omega}'_y \\ \dot{\omega}'_z \end{bmatrix} = \begin{bmatrix} J'^{-1}_{xx}\left(n'_x - \omega_y\omega_z(J'_{zz} - J'_{yy})\right) \\ J'^{-1}_{yy}\left(n'_y - \omega_x\omega_z(J'_{xx} - J'_{zz})\right) \\ J'^{-1}_{zz}\left(n'_z - \omega_x\omega_y(J'_{yy} - J'_{xx})\right) \end{bmatrix} \tag{II.3.11}$$

II.3.3 Modelling the Robot

We shall now define the model of the walking robot, as shown in Figure II.3.3, which is similar to that of Ho et al. [50], but with 3 DOF per leg. The robot is modelled as a single body with mass m acting vertically downwards in the ground coordinate system from its centroid. There are four spherical joints, four revolute joints and four feet to consider as nodes in the system. Simulating the limbs of the robot as rigid bodies connecting these nodes, and the body as a flat plane between the four cylindrical joints, it is sufficient

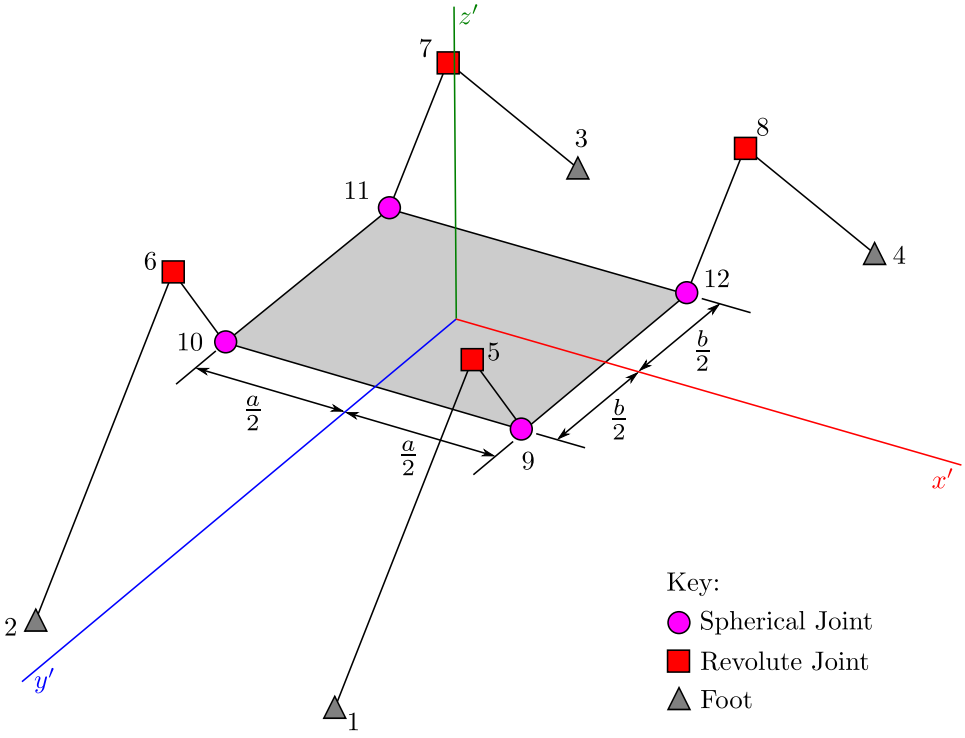


Figure II.3.3: Model of the Quadruped Robot

to consider these nodes solely when it comes to calculating floor collisions and reaction forces. In fact, within the leg angle limits it is not possible for the knee joint in each leg to contact the ground plane without either the hip or the foot penetrating the floor, unless the robot is inverted. There are therefore eight conditional vertical reaction forces which may apply to the body depending on which of these nodes are in contact with the ground.

The leg model is the same as Section II.3.1, and we can use Equations II.3.1, II.3.2 and II.3.3. Including the offsets to the origin of the body reference frame, where the coordinates for each foot f are $\mathbf{s}'_f = \left[x'_f, y'_f, z'_f\right]^T$ for $f \in [1, 2, 3, 4]$, in accordance with the arrangement set out in Figure II.3.4,

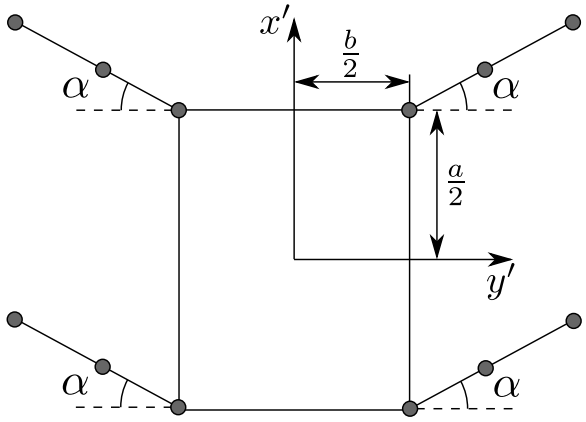


Figure II.3.4: Orientation of Feet Coordinate Systems

then:

$$\begin{bmatrix} \mathbf{s}'_1 \\ \mathbf{s}'_2 \\ \mathbf{s}'_3 \\ \mathbf{s}'_4 \end{bmatrix} = \begin{bmatrix} \frac{a}{2} + \mathrm{s}(\alpha_1)\left(l_1\mathrm{c}(\beta_1) + l_2\mathrm{c}(\gamma_1 - \beta_1)\right) \\ \frac{b}{2} + \mathrm{c}(\alpha_1)\left(l_1\mathrm{c}(\beta_1) + l_2\mathrm{c}(\gamma_1 - \beta_1)\right) \\ l_1\mathrm{s}(\beta_1) - l_2\mathrm{s}(\gamma_1 - \beta_1) \\ -\frac{a}{2} + \mathrm{s}(\alpha_2)\left(l_1\mathrm{c}(\beta_2) + l_2\mathrm{c}(\gamma_2 - \beta_2)\right) \\ \frac{b}{2} + \mathrm{c}(\alpha_2)\left(l_1\mathrm{c}(\beta_2) + l_2\mathrm{c}(\gamma_2 - \beta_2)\right) \\ l_1\mathrm{s}(\beta_2) - l_2\mathrm{s}(\gamma_2 - \beta_2) \\ -\frac{a}{2} + \mathrm{s}(-\alpha_3)\left(l_1\mathrm{c}(\beta_3) + l_2\mathrm{c}(\gamma_3 - \beta_3)\right) \\ -\frac{b}{2} - \mathrm{c}(\alpha_3)\left(l_1\mathrm{c}(\beta_3) + l_2\mathrm{c}(\gamma_3 - \beta_3)\right) \\ l_1\mathrm{s}(\beta_3) - l_2\mathrm{s}(\gamma_3 - \beta_3) \\ \frac{a}{2} + \mathrm{s}(-\alpha_4)\left(l_1\mathrm{c}(\beta_4) + l_2\mathrm{c}(\gamma_4 - \beta_4)\right) \\ -\frac{b}{2} - \mathrm{c}(\alpha_4)\left(l_1\mathrm{c}(\beta_4) + l_2\mathrm{c}(\gamma_4 - \beta_4)\right) \\ l_1\mathrm{s}(\beta_4) - l_2\mathrm{s}(\gamma_4 - \beta_4) \end{bmatrix} \tag{II.3.12}$$

Where the abbreviations c = cos and s = sin are used for brevity.

We must also calculate the velocities of each foot relative to the body fixed reference frame ($\dot{\mathbf{s}_f}'$), which can then be related to the ground reference frame ($\dot{\mathbf{s}_f}$). These are easiest found numerically by calculating the differences between the values at the previous and current time steps:

$$\dot{\mathbf{s}_f}' \approx \frac{\mathbf{s}'^{t}_f - \mathbf{s}'^{t-dt}_f}{dt} \tag{II.3.13}$$

II.3.4 Forces on the Body

To cover all cases, we must not only consider the forces on the feet of each leg and the centroid of the body, but also the hip nodes (numbered 9, 10, 11 and 12 on Figure II.3.3), as they may contact the ground ahead of the corresponding foot. Nodes which are not in contact with the ground shall not be subject to any forces, only a constant body force due to acceleration by gravity ($g \approx -9.81\text{ms}^{-1}$) will apply.

II.3.4.1 Ground Reaction

When a body is resting on the ground and subject to gravity it will experience a vertical reaction force in the ground frame ($R = mg$). The body contact points will experience reaction forces, and for static equilibrium the sum of forces and moments must be zero ($\sum \mathbf{F} = 0$ and $\sum \mathbf{N} = \sum \mathbf{s} \times \mathbf{F} = 0$). This forms a set of six equations which can be solved for up to six unknowns. However, for many common scenarios the configuration of reaction forces is indeterminate [81] and must be found through a numerical scheme or Linear Complementary Problem (LCP) which is computationally expensive.

Most often in a numerical simulation with a constant timestep collisions will not take place at exact multiples of Δt and so the simulation will overshoot slightly. The smaller the time constant the less this will be, but a traditional method of dealing with this is to introduce a penalty force based upon the degree of interpenetration [116]. The ground is considered as a combined spring-damper system and the penalty force determined as follows:

$$f_p = k_s d\hat{\mathbf{n}} - k_d(\dot{\mathbf{r}}.\hat{\mathbf{n}})\hat{\mathbf{n}} \tag{II.3.14}$$

Where k_s is the spring coefficient, k_d is the damping coefficient, d is the

distance of interpenetration, $\dot{\mathbf{r}}$ is the velocity of the point and $\hat{\mathbf{n}}$ is the normal vector of the penetrated surface. This requires a large spring coefficient, the determination of which has been called a 'black art' [81], which can lead to instability and is often only suitable for a particular simulation scenario. A refined version of this model incorporates an integral term over the period of contact to eliminate any steady state error (i.e. the constant interpenetration for the 'spring' to balance the body) [32]. A factor λ is used to prevent any unwanted 'popping'(a term denoting where in a single timestep an object penetrates the ground sufficiently that the restitution forces cause it to 'pop' out of the ground) and is usually in the range $0.8 < \lambda < 0.9$:

$$f_p = \max\left(\frac{k_s d\hat{\mathbf{n}} - k_d(\dot{\mathbf{r}}.\hat{\mathbf{n}})\hat{\mathbf{n}} + k_i I}{n_c}, 0\right) \quad \text{(II.3.15)}$$

$$I = \sum_{t_c} \lambda d \quad \text{(II.3.16)}$$

Here, k_i is the integral constant.

An alternative approach is to perpetuate static contact by a series of chained micro-collisions [81]. This is modelled by artificially increasing the coefficient of restitution up to a maximum value in the impulse calculation introduced in Section II.3.4.2 when a collision is designated a micro-collision. The test condition is if:

$$-\dot{r}_z < \sqrt{(2g\eta_c)} \quad \text{(II.3.17)}$$

Where η_c is the maximum distance between objects said to be in static contact.

In our case, application of the penalty based method has been found to lead to the most stable results.

II.3.4.2 Collision

There are many approaches used to simulate the collision of objects, and there exists a vast body of literature within both traditional mechanics and

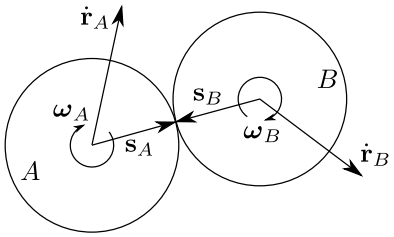


Figure II.3.5: Collision of Two Bodies

computer science [6]. The two main approaches applicable here can be considered to fall under constraints [48] or impulse based mechanics. The object of much of the simplification undertaken herein is to remove the need to solve constraint equations and so we shall focus upon the significantly simpler impulse approach [81], using the form of equations developed by Vella [116].

Upon the instant of contact, the body will undergo a rapid deceleration and corresponding change in momentum resulting in an impulse. We require that the foot comes to a vertical rest in one time step (Δt), so that it does not pass through the ground plane, so:

$$I = m\dot{r} = F\Delta t \tag{II.3.18}$$

$$F = \frac{m\dot{r}}{\Delta t} \tag{II.3.19}$$

More complex however is the treatment of angular momentum L which is defined for an object with fixed mass rotating about a fixed symmetrical axis as $\mathbf{L} = \mathbf{J}\boldsymbol{\omega}$, whereas, for a particle about an origin $\mathbf{L} = \mathbf{r} \times \mathbf{p}$, where $\mathbf{r}$ is the position vector of the particle and $\mathbf{p}$ is its linear momentum.
These can be combined to calculate the total impulse including the changes in both linear and angular momentum [116]. For two bodies colliding, as shown in Figure II.3.5, the magnitude of the impulse I is:

$$\frac{-(1+\epsilon)\dot{\mathbf{r}}^{AB}.\mathbf{n}}{M_A^{-1}+M_B^{-1}+\left[\left(\mathbf{J}_A^{-1}(\mathbf{s_A}\times\mathbf{n})\right)\times\mathbf{s}_A+\left(\mathbf{J}_B^{-1}(\mathbf{s_B}\times\mathbf{n})\right)\times\mathbf{s}_B\right].\mathbf{n}} \tag{II.3.20}$$

Where M is the mass of the body, $\mathbf{I}$ its moment of inertia, ϵ the coefficient

of restitution and $\dot{\mathbf{r}}^{AB}$ the relative velocity between the bodies at the contact point. This can be simplified for our case, as we know that we are coming solely in contact with the ground which is flat in the (x,y)-plane. As the mass of the ground and its inertia can be considered to be infinite and the normal vector is simply vertical in the ground plane this becomes:

$$I = \frac{-(1+\epsilon)\dot{\mathbf{r}}_f.\mathbf{k}}{m^{-1} + \left[(\mathbf{J}^{-1}(\mathbf{s}_f \times \mathbf{k})) \times \mathbf{s}_f\right].\mathbf{k}} \tag{II.3.21}$$

Where $\dot{\mathbf{r}}_f$ is the velocity of the foot, $\mathbf{s}_f$ is the position of the foot, m is the mass of the body, $\mathbf{J}$ is its moment of inertia and $\mathbf{k}$ is the unit vector in the z-direction of the ground plane. This impulse magnitude can be transformed into a force using Equations II.3.28 and II.3.29 and applied in the EOM. All of these quantities are required in the ground plane, and therefore require the transformation of both the foot velocity and body moment of inertia into the global frame using Equations V.C.4 and V.D.7.

The impulse expression, given in Equation II.3.21, is valid for a single point of contact. If there are multiple contact points, one cannot simply repeat the calculation for each, as it does not take into account the support of the other contact points and would be akin to simulating a universal joint at the COM rather than a rigid body. The impulse must therefore be applied at the average point of contact, but calculated for each point separately to account for their relative speeds and facilitate the evaluation of leg friction forces. We therefore define the following expression to be evaluated for each contact point, where the number of such points is n_c.

$$I_f = \frac{-(1+\epsilon)\dot{\mathbf{r}}_f.\mathbf{k}}{n_c\left(m^{-1} + \left[(\mathbf{J}^{-1}(\mathbf{s}_c \times \mathbf{k})) \times \mathbf{s}_c\right].\mathbf{k}\right)} \tag{II.3.22}$$

Where:

$$\mathbf{s}_c = \frac{1}{n_c}\sum_i^{n_c} s_i = \mathbf{A}\left(\frac{1}{n_c}\sum_i^{n_c} s_i'\right) \tag{II.3.23}$$

This formulation of the expressions allows for a simple one-step solution which fits neatly into the simulator, but is only valid for planar surfaces.

This however is in-line with the initial assumptions made. Our initial focus is in the developments of gaits for simple robots in simple environments to evaluate the efficacy of a range of neuron models and techniques for encoding delays. Further to this work, should the results in this Thesis warrant continued investigation, a more sophisticated environment may well be advisable for the inclusion of complex terrain geometry.

II.3.4.3 Friction

When a foot is in contact with the ground and attempts to slide across the ground plane a frictional force is experienced. In this way the feet of the robot gain traction and (with an appropriate gait) move the body of the robot forwards. The most popular, and simplest, model is Coulombs Law of Friction governed by the equation:

$$F_{fr} \leq \mu F_n \tag{II.3.24}$$

Where F_n is the normal force, μ is the coefficient of friction and F_{fr} is the force of friction parallel to the surface opposing the direction of applied force. In vector form this can be written:

$$\mathbf{F}_{fr} = -\mu|\mathbf{F}_n|\frac{\dot{\mathbf{r}}_{x,y}}{|\dot{\mathbf{r}}_{x,y}|} \tag{II.3.25}$$

Where the contact point is moving $\dot{\mathbf{r}}_{x,y}$ in the (x,y) plane. The friction force will balance any applied force up until it exceeds $\mu_s\mathbf{F}_n$ (where μ_s is the static coefficient of friction), whereupon it is limited to $\mu_k\mathbf{F}_n$ (where μ_k is the kinetic coefficient of friction). μ_k is most often less than μ_s and values vary widely depending on the configuration of materials used [80]. This may be incorporated into simulation, which already uses penalty based ground reactions, by employing a virtual spring approach [116], as shown in Figure II.3.6. When the point makes contact with the ground, this point is stored and a virtual spring between this point ($\mathbf{a}_0$) and the current position

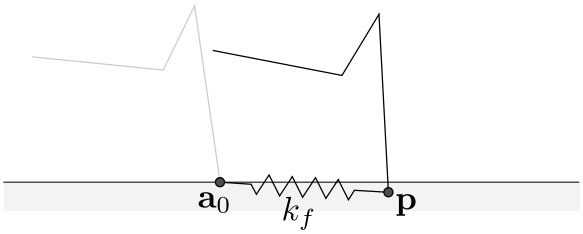


Figure II.3.6: Penalty Friction Model

of the point ($\mathbf{p}$) applies a restorative force limited to $\mu_s \mathbf{F}_n$. If this limit is reached the friction enters a dynamic mode and is constant ($\mu_k \mathbf{F}_n$) in the direction of the spring.

$$\mathbf{F}_s = k_f(\mathbf{p} - \mathbf{a}_0) \tag{II.3.26}$$

$$\mathbf{F}_{fr} = \begin{cases} \mathbf{F}_s & \text{if } |\mathbf{F}_s| \leq \mu_s |\mathbf{F}_n| \\ \mu_k |\mathbf{F}_n| \frac{\mathbf{p}-\mathbf{a}_0}{|\mathbf{p}-\mathbf{a}_0|} & \text{else if } |\mathbf{p} - \mathbf{a}_0| \neq 0 \\ \mathbf{0} & \text{else} \end{cases} \tag{II.3.27}$$

Where k_f is the virtual spring coefficient and $\mathbf{F}_s$ is the virtual spring force vector.

II.3.5 Bringing it all together

There exist a set of f 3D force vectors, assembled from each of the possible ground contact points (namely the foot and hip of each of the four legs of the robot). The overall force applied to the body modelling the robot is a linear summation of these terms. The torque vector applied to the body can then be calculated from these forces and our knowledge of their application points in the body reference frame.

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & mg \end{bmatrix}^T + \sum_f \mathbf{F}_f \tag{II.3.28}$$

Where:

$$\mathbf{F}_f = \begin{cases} \begin{bmatrix} \mathbf{F}_{fr}.\mathbf{i} & \mathbf{F}_{fr}.\mathbf{j} & \frac{1}{\Delta t} I + f_p \end{bmatrix}^T & \text{if} z_f \leq 0 \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T & \text{else} \end{cases} \tag{II.3.29}$$

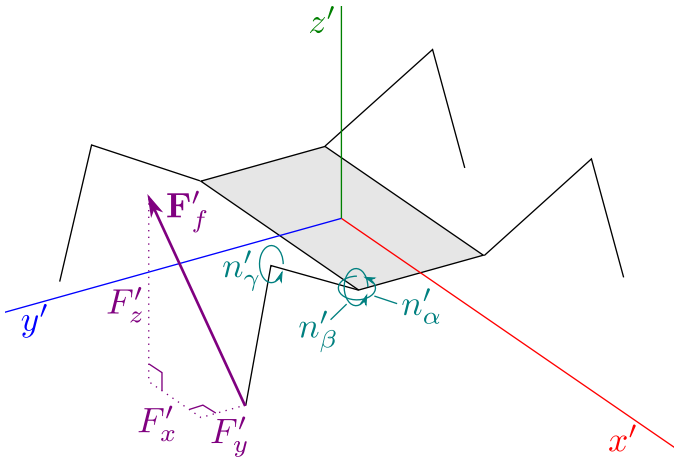
The overall torque applied to the body is the sum of all foot forces in the body frame multiplied by the appropriate moment arms.

$$\mathbf{n}' = \sum_f \mathbf{n}'_f = \sum_f \left(\mathbf{s}'_f \times \mathbf{F}'_f\right) = \sum_f \left(\mathbf{A}^T\mathbf{s}_f \times \mathbf{A}^T\mathbf{F}_f\right) \qquad \text{(II.3.30)}$$

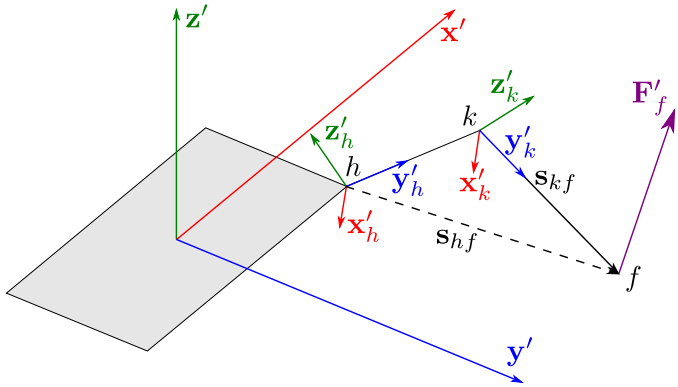
The system under consideration is highly non-linear, energy dissipative and discontinuous; therefore it is impossible to solve explicitly and we must seek a numerical approximation to the solution. Our structuring of the system of equations has very deliberately eliminated the need for complex treatment such as elimination and factorisation techniques. Instead we can simply integrate Equations II.3.10 and II.3.11 using one of the many methods for numerically integrating ordinary differential equations [2].

II.3.6 Servo Torques and Body Inversion

Whilst we have assumed that limb movements are absolute, we are ultimately interested in real robots and so it is useful to be able to check gaits to see if they are realisable. Standard servos receive an absolute angular demand as a Pulse Width Modulation (PWM)) signal which makes them ideal for robotics applications. They have a maximum speed at which they will rotate angularly (quoted in s/60° with no-load) and a maximum torque (usually given in kgcm). These values vary greatly depending on the materials and quality of construction, with currently available servos developing torques varying from around 0.1Nm to 10Nm and speeds from 0.1s/60° to 0.3s/60°. The demand rotational speed is easily found by inspecting the specific gait algorithm under investigation, and the torque at the hip joint required to react the forces at the foot can be found as follows. Due to our simplifying assumption of a single body for the robot, we do not have a rotation matrix for each limb requiring that we must fall back on more traditional trigonometry and vector algebra to calculate the torques at each of the required joints. Consider the first leg made up of nodes 1,5 and 9.



(a) Leg Servo Torques



(b) Limb Coordinate Systems

Figure II.3.7: Checking Leg Servo Torques

This can be actuated using three servos, one providing a torque around the z'-axis at node 9 giving rise to the leg angle α. The second servo must also act at node 9 normal to the plane of the leg driving β. Finally, a servo acts at the knee, node 5, once again normal to the leg plane to drive γ. These three torques are denoted n'_α, n'_β and n'_γ respectively and are shown in Figure II.3.7.

The magnitude of the joint torque is equal to the magnitude of the component of the moment interaction between the two links in the rotational direction [35]. By using a Free Body Diagram approach we may arrive at the following identities for the joint torques [73, 7] using the nomenclature indicated in Figure II.3.7.

$$\begin{bmatrix} n'_\alpha \\ n'_\beta \\ n'_\gamma \end{bmatrix} = \begin{bmatrix} (\mathbf{z}' \times \mathbf{s}'_{hf})^T \\ (\mathbf{x}'_h \times \mathbf{s}'_{hf})^T \\ (\mathbf{x}'_k \times \mathbf{s}'_{kf})^T \end{bmatrix} \mathbf{F}'_f \tag{II.3.31}$$

This leaves us to find $\mathbf{x}'_h = \mathbf{x}'_k$ which can be found as it is the unit normal vector to the two limbs, thus:

$$\mathbf{x}'_h = \mathbf{x}'_k = \frac{\mathbf{s}'_{hk} \times \mathbf{s}'_{kf}}{|\mathbf{s}'_{hk} \times \mathbf{s}'_{kf}|}. \tag{II.3.32}$$

In this way, the torque which is required of the servos to develop the motion witnessed in the simulation may be found. As with checking for robot inversion, our key focus is on our ability to realise such systems in the real world and to minimise the cost of simulation.

The aim of developing this model is to enable the search for stable gaits, and is not concerned with self-righting (although the simulation environment could certainly be used as such if desired). Especially considered in an ER context, we wish to end a trial simulation as soon as we are sure that the gait is not valid. To this end it is useful to be able to determine if the robot has inverted, i.e. the vertical component of the body normal vector $\mathbf{z}'$ in the ground frame going below zero. So as not to unfairly dismiss potentially acrobatic gaits, we can check for $z_z < 0$ when any contact is detected.

It is worth noting the constraint on the Euler parameters that $\mathbf{p}^T\mathbf{p} = 1$ and how this holds true when the simulation relies on a numerical integration scheme with error. This constraint can be visualised as a 4D unit sphere, where $\mathbf{p}$ must lie on the surface. When the EOM are integrated numerically the new $\mathbf{p}$ will move along a tangent away from the surface violating the constraint. To correct for this, we can scale the new $\mathbf{p}$ to ensure the constraint holds by recognising that $\mathbf{p}^T\mathbf{p} = \mathbf{p}^2$. Equation II.3.33 shows the scaling relationship which will always ensure the constraint is satisfied and that the relationship between $e_0,\ldots,e_3$ remains the same:

$$\mathbf{p} = \frac{\mathbf{p}}{\sqrt{\mathbf{p}^T\mathbf{p}}} \tag{II.3.33}$$

Clearly the fact that the numerical integration has not rotated the body around the surface of the $\mathbf{p}^T\mathbf{p}$ sphere means that an error has been introduced, but this is a necessary evil of such a scheme and in most scenarios this error is very small. In fact $\mathbf{p}^T\mathbf{p}$ can be a good metric to observe during the simulation, as if it approaches $\mathbf{p}^T\mathbf{p} \approx 1.1$ very high rotational speeds must render the development of a worthwhile gait highly unlikely.

II.3.7 Solution Procedure

We can now bring all of this work together and define a step-by-step procedure to solve these equations many times over for the duration of the desired simulation.

1. Define the physical constants affecting the simulation, μ, ϵ, δt, g, k_s, k_d, k_p, k_i, and λ.

2. Define the geometry of the body l_1, l_2, a, b, c and Γ as per Figures II.3.2 and V.D.1.

3. Initialise mass m and inertial matrix $\mathbf{J}$ using Equations V.D.1, V.D.2, V.D.3, V.D.4 and V.D.5.

4. Set initial configuration of legs (α, β, γ), and body orientation ($\mathbf{p} = [1, 0, 0, 0]^T$, $\dot{\boldsymbol{\omega}}' = \mathbf{0}$). Also set the initial position, velocity and acceleration of the robot.

5. For all timesteps in simulation:

   (a) Calculate driving angles (α, β, γ) based on the desired gait and their derivatives $(\delta\alpha, \delta\beta, \delta\gamma)$ for each leg.

   (b) Calculate leg positions and velocities using Equations II.3.12 and II.3.13.

   (c) Calculate $\mathbf{A}$ from $\mathbf{p} = [e_0, \mathbf{e}]^T$ using Equation V.C.8, $\mathbf{G}$ from Equation V.C.11, $\dot{\mathbf{A}}$ with Equation V.C.3 and $\mathbf{J}^{-1}$ using Equation V.D.7.

   (d) Determine which nodes (if any) are in contact with the ground using the conditions given in Equation II.3.28 and average their positions to find $\mathbf{s}_c$ and update the integral term of Equation II.3.16. If this marks the start of a contact phase, store the anchor point $\mathbf{a}_0$.

   (e) For each contact point:

      i. Transform the foot position $\mathbf{r}_f$ and velocity $\dot{\mathbf{r}}_f$ into the global frame using Equations V.C.1 and V.C.4.

      ii. If $\dot{\mathbf{r}}_{f_z} < 0$, calculate the impulse using Equation II.3.22.

      iii. Find the friction force using Equation II.3.27.

      iv. Calculate the penalty forces, as per Equation II.3.16.

   (f) Sum individual node forces to obtain the total body force $\mathbf{F}$ using Equations II.3.28 and II.3.29.

   (g) From the individual forces, calculate the torque on the body ($\mathbf{n}'$) using Equation II.3.30.

(h) If required, calculate the servo demand torques using Equations II.3.31 and II.3.32.

(i) Solve the EOM, Equations II.3.10 and II.3.11, in terms of $\ddot{\mathbf{r}}$ and $\dot{\boldsymbol{\omega}}'$.

(j) Integrate $\dot{\boldsymbol{\omega}}'$ to $\boldsymbol{\omega}'$. Transform $\boldsymbol{\omega}'$ into $\dot{\mathbf{p}}$ using Equation II.3.7.

(k) Numerically integrate $\ddot{\mathbf{r}}$ and $\dot{\mathbf{p}}$ to find the updated position and orientation of the body. In our case we shall use the standard Fourth Order Runge-Kutta method.

(l) Normalise the Euler parameter to ensure that $\mathbf{p}^T\mathbf{p} = 1$ despite errors in numerical integration using Equation II.3.33.

(m) Increase simulation time by Δt and proceed to the next timestep.

II.3.8 Realism and Computational Efficiency

The formulation and assembly of these algorithms make significant simplifying assumptions about the geometry of the robot, but the equations of motion are solved directly. It is only in the treatment of ground reactions and friction where we have to rely on approximations. These nevertheless provide a simple, computationally minimalistic approach which still realistically models static and dynamic friction. It also finds appropriate ground reactions in a single step without resorting to an LCP method. In this way we aim to strike a balance between the 'is right' view of traditional physical simulation and the 'looks right' approach of games simulation. The majority of numerical physical simulations incorrectly model impacts with a low coefficient of restitution.

It is somewhat difficult to compare the computational efficiency of different simulation engines given the incredible variety of simulation approaches, integration methods, object representations, constraints and platforms available. Comparing like with like is practically impossible and all we can

reasonably do is make some judgements about the degree of computation undertaken during a simulation and the accuracy with which it does it. However, any code which is optimised for a single purpose, without any of the extraneous considerations which more general purpose environments must have, should be faster. Specifically, the algorithms described herein have a significantly reduced set of ODEs to solve, which have easily determined solutions. There is not need to solve LCP problems for constraints and ground reactions. Because the kinematics of the limbs are determined geometrically and their speeds are found numerically, we are solving a one body problem instead of a nine body problem. However, without extensive comparison no real conclusions can be made. What we can be sure of however is that by following the long established protocol within ER for designing minimalistic simulations which simply capture the essential behaviour under investigation, that a simulation environment has been developed which is fit for the purpose of evolving quadruped locomotory gaits.

II.3.9 Simulation Results

To test the minimalistic simulation presented here a (1,3,4,2)[1] crawling tetrapod gait was designed with leg angles as shown in Figure II.3.8, a duty cycle of 0.75 and implemented in the simulation environment. For each swing phase the leg angle α is designed to achieve a constant linear motion in the x-direction, with the corresponding β and γ joint angles altering to maintain a constant body height and y-axis foot contact point.

To achieve a constant forward speed, the distance moved along the x-axis by all feet in contact with the ground (dx) per time step should be the same throughout the gait. Through trigonometric analysis we can derive an expression for the leg angle at the next time step, shown in Equation II.3.34 below. For these expressions an assumption of equal leg lengths ($l_1 \equiv l_2 \equiv l$)

[1] Using the leg labelling nomenclature given in Figure II.3.3

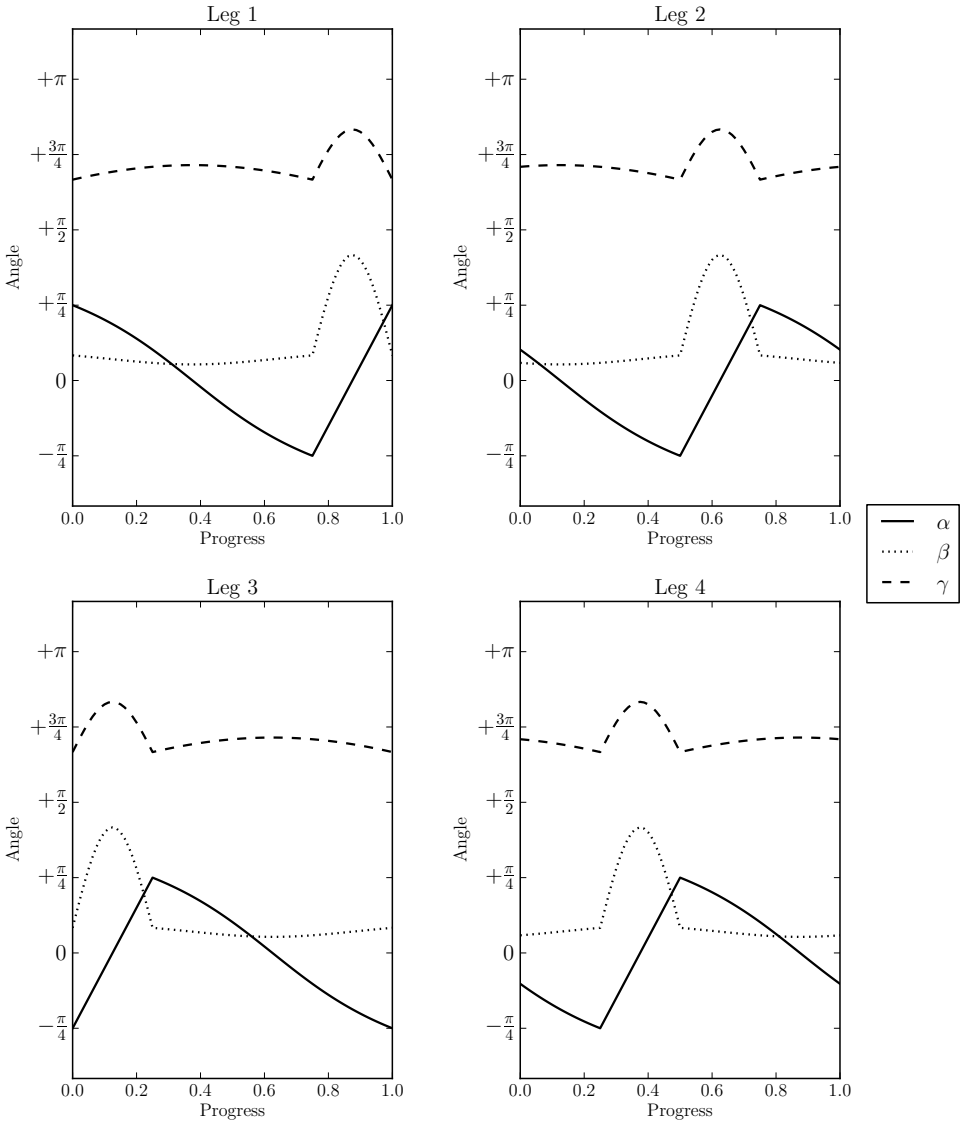


Figure II.3.8: Gait Leg Angles

significantly simplifies both the derivation and the results to an extent which renders them solvable.

$$\alpha_{t+1} = \text{atan}\left(\tan\alpha - \frac{dx}{r}\right) \tag{II.3.34}$$

Where, for initial leg angles α_0, β_0 and γ_0, r is as per Equation II.3.39 and a gait evaluated for n steps we can then find dx as below:

$$dx = \frac{2l\sin\alpha_0(\cos\beta_0 + \cos(\gamma_0 - \beta_0))}{n-1} \tag{II.3.35}$$

Our aim is to eliminate any foot slippage and ensure a constant body height throughout the gait, and therefore the leg angles β and γ must alter throughout each swing phase to ensure that the foot y-axis and z-axis position remains constant. If the following expressions are true:

$$2lr + r^2 + z^2 \neq 0 \tag{II.3.36}$$

$$r^2 + z^2 \neq 0 \tag{II.3.37}$$

$$l\left(z\sqrt{(r^2+z^2)(4l^2-r^2-z)^2} + 2lr^2 + 2lz^2 + r^3 + rz^2\right) \neq 0 \tag{II.3.38}$$

Where:

$$r = \frac{l\cos\alpha_0\left(\cos\beta_0 + \cos(\gamma_0 - \beta_0)\right)}{\cos\alpha} \tag{II.3.39}$$

$$z = l\left(\sin\beta_0 - \sin(\gamma_0 - \beta_0)\right) \tag{II.3.40}$$

Then:

$$\beta = 2\text{atan}\left(\frac{\sqrt{(r^2+z^2)(4l^2-r^2-z^2)} + 2lz}{2lr + r^2 + z^2}\right) \tag{II.3.41}$$

$$\gamma = 2\text{atan}\left(\frac{\sqrt{(r^2+z^2)(4l^2-r^2-z^2)}}{r^2+z^2}\right) \tag{II.3.42}$$

Figure II.3.8 graphically presents the evaluation of these expressions for the four legs with $\alpha_0 = \frac{\pi}{4}$, $\beta_0 = \frac{\pi}{9}$ and $\gamma_0 = \frac{4\pi}{6}$. For the return phase, the leg is raised and swung back to α_0 in a sinusoidal profile. This walking gait is statically stable as supporting tripods contain the centroid of the body in

Symbol	Value	Symbol	Value	Symbol	Value
μ	0.6	k_d	5.0	l_1	0.100 m
ϵ	0.1	k_i	100.0	l_2	0.125 m
δt	0.0001s	k_f	10000.0	a	0.20 m
g	$-9.81\ \mathrm{ms}^{-2}$	λ	0.6	b	0.10 m
k_s	10000.0	γ	$250.0\ \mathrm{kgm}^3$	c	0.10 m

Table II.3.1: Simulation Constants

all steps, and the results presented are for a step time of 3.0s. These results incorporate a half second settling time, waiting for the feet of the robot to interpenetrate the ground under the action of gravity and stably establish the correct reaction forces before the gait commences. The simulation constants were set as shown in Table II.3.1 and it was run for 15 seconds. The penalty spring constant k_s was set to allow the body to penetrate the ground under 1mm when all four legs support the static body ($k = \frac{n_c mg}{s}$). The values for μ and ϵ are realistic, with k_i and λ manually designed to quickly eliminate any steady state error whilst avoiding 'popping'. The penalty friction spring constant k_f was set high to ensure that the foot could not move very far without transitioning between the static and dynamic behaviour. The exact distance is not unique and is dependent on the reaction experienced by the foot in question.

A quadruped robot was built to match the parameters of the simulation given in Table II.3.1 in order to allow physical verification of both the gait and the accuracy of the simulation developed; a photograph of the robot is shown in Figure II.3.1. The design and assembly of the robot is described in detail in Chapter II.4.1. Clearly, the legs of the robot form a considerable part of the mass of the robot, as the servo motors are mounted on the axis of each joint in the common manner. Whilst the structure appears somewhat massive, the foam cored plastic construction meant that the mass

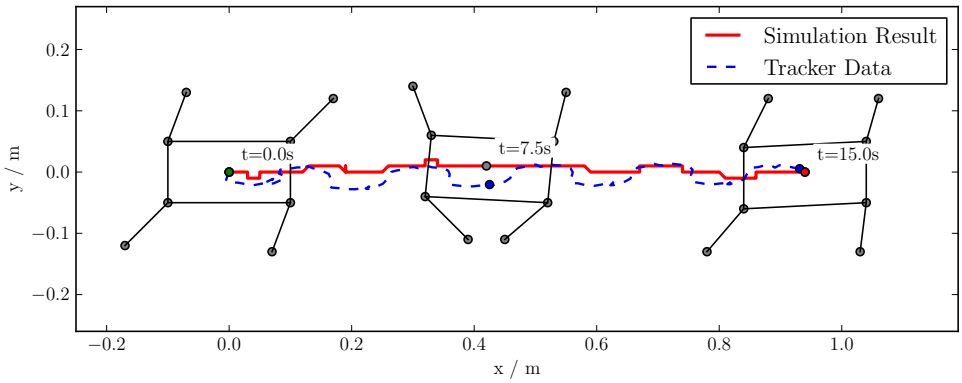


Figure II.3.9: Top View of Simulation Result Overlayed with Real Data From Laser Tracker

was minimal with the majority of the leg weight contributed directly from the motors. The hip servo (α and β) motors are mounted at each of the four corners of robot and contribute equally to the mass, location of the COM and inertia of the body. As their mass and inertial properties are almost invariant with respect to the angle of the legs (due to the very small moment arm around the joint centres), they may be considered for all intents and purposes to form part of the body and not the legs. The knee (γ) motor is of the 9g servo variety and thus the motors in the legs form around 7% off the mass of the robot without batteries.

The robot used low cost motors, eight standard servos (for α and β leg angles) and four micro servos (for γ leg angles) throughout and was controlled by an Arduino[2] based microcontroller. Power was supplied from a 7.2V 1800mAh Nickel-Cadmium (Ni-Cd) battery through a Battery Elimination Circuit (BEC) which supplied constant 6V to the servos with a limited current draw of 5A. The microcontroller was slaved to a personal computer over a Universal Serial Bus (USB) to serial link and the gait used in the simulation replicated. The movement of the robot was recorded using a state-of-the-art Leica LTD 500 laser tracker following a 1.5" spherical

[2]www.arduino.cc.

reflector, which can be seen in Figure II.3.1, with a high degree of accuracy.

This robot has only one purpose, which is to support the validation of the simulation and attest to the comparability of results obtained virtually with what one might expect should the gait be implemented in hardware. It is not intended that evolved gaits are trialled on the real robot, but that the simulation is used solely to verify their performance. The results of evolutionary optimisation of such gaits are presented in Chapter III.1.

The overhead trajectory can be seen in Figure II.3.9 and shows how the gait moves the body of the robot forwards in a straight line with an excellent degree of agreement between the simulated and real results. The simulation result shows a slight undulatory profile which is matched, and exaggerated, by the real trajectory. There are a number of contributory factors to this deviation. First, the reflector was positioned around 100mm above the plane of the hip nodes and any tilting will increase any observed y-axis deviations in the robot's path. Secondly, the low-cost servos used were imperfect and operating near the limit of their capability and finally the not-insignificant mechanical play present in the leg linkages added some sprawl to the robot's stance. Despite this the key features of the gait are reproduced in both data sets and, most importantly, the predicted distance travelled is accurate to within a few centimetres. This is acceptable for our purposes as the simulated gait very closely resembles the real robot motion, and so the simulator can be used to predict distances travelled for the measurement of fitness. Whilst it is beneficial for a high degree of accuracy between the two, in reality all that is required is that a better controller travels proportionally further and thus achieves a higher fitness proportional to its superiority.. This gait has a Froude number ($\frac{u^2}{gh}$, where u is the characteristic speed, g is the acceleration due to gravity, and h is the standing height of the hip) of approximately 8.2×10^{-3} [1]. For the duration of the simulation the torque at each hip is shown in Figure II.3.10. Each torque point shown is

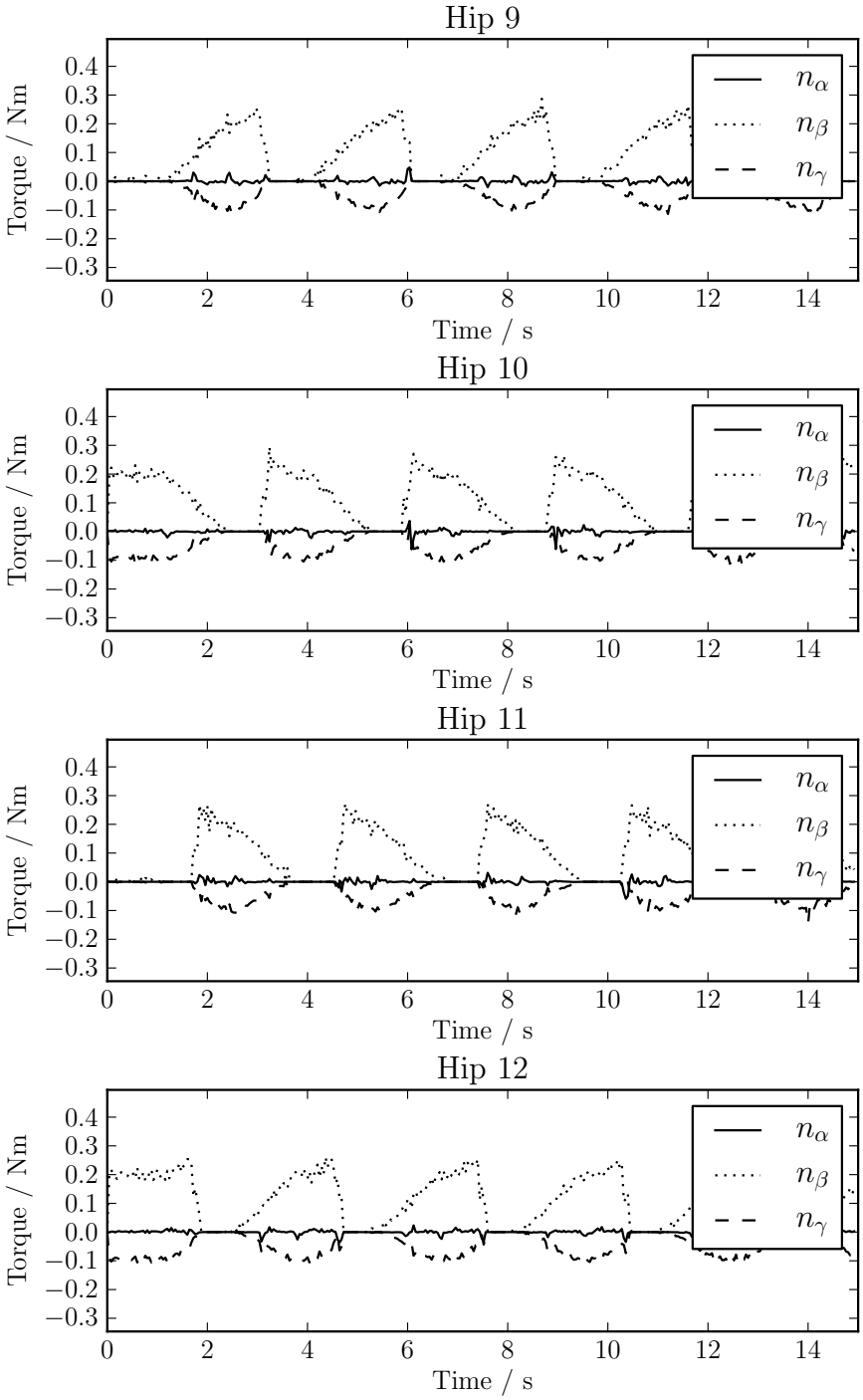


Figure II.3.10: Simulated Servo Torques During the Simulation

an average taken over a period of 0.1s. As the limbs are considered to be massless, there is no torque required to move them, explaining the 0.75s long gaps in the torque curves (leg raise, swing and then lower). The β leg angle requires the highest torque across all legs as it is responsible for keeping the robot off the ground. The sloped nature of the β torque profile during stance phases clearly show how the COM shifts relative to each planted foot due to the constant speed gait. As more weight is reacted by feet closer to the COM the required torque is greater. One can recognise staggered similar patterns in the torque curves as the gait progresses, with the torque required limited to under 0.3Nm, which is easily realisable using common model servos. The gait presented here is somewhat conservative and for a more dynamic version, moving towards trotting or galloping gaits, the torque requirements are likely to be significantly increased for transitory impacts.

Due to the simplicity of the robot developed, there is no telemetry to validate the prediction of leg torques provided by the simulation. Whilst they appear to take the appropriate qualitative form, with the shifting centre of mass seen clearly in the torque plot, it is impossible to gauge their accuracy. For one thing, as the real robot's limbs have a mass which forms a significant proportion of the whole, the torque required for limbs not in contact with the ground will be non-zero. However, we can state that as the gait was successfully achieved that the torque requirements must be less than the maximum torque which the servos can provide. This is limited to 3.7 kgcm (0.37 Nm) at 6V which is the maximum safe voltage for the motors. This is also only every so slightly greater than the peak torque observed during the simulated gait for the β leg angles, with α and γ requiring significantly less.

Whilst we cannot strongly claim that the physical results presented completely validate the simulation, at the very least we can state that the simulation correctly represents the modelling of friction and stability. The key

focus of this work was to develop a tool for the evaluation of evolved gaits. We can see how the simulation was extremely accurate in the estimation of the distance travelled by the real robot. Alongside this, we are most interested in the stability of the robot during its gait. This is more significant as the number of redundant limbs reduces. Beer and Gallagher adopted a 2D simulator for a hexapod walker, which simply considered the robot having a fixed location when incorrectly supported, but a 6 legged robot is significantly more stable than a quadruped. With fewer, and longer, legs the chances of recovering from a significant instability are small, emphasising the need for such analysis.

II.3.10 Extensions

Possibly the most natural and simplest of extensions to this simulation model is to move to a different leg posture or a hexapod robot, which requires only the trivial extension of the appropriate feet coordinates in Equation II.3.12 to the new body geometry. The hexapod legged robot is very common in studies of ground locomotion as it is more stable than that of its quadruped cousins and simpler to design. The additional computational expense is minimal and the algorithm remains the same. Moving to a different geometrical arrangement of limbs would require only some reworking of the trigonometric identities governing the body frame position of the feet.

The robot described here is modelled as entirely rigid. To increase realism and add useful damping for more energetic gaits, it may be desirable to add highly damped torsion springs at each servo to 'soak up' some of the impact forces. This would average them over an increased period of time, decreasing the contact forces and giving the servos an easier time whilst modelling the play present in most mechanical linkages. Whilst this model allows for the checking of the torques required of the servos at each joint, it would be useful to set a limit after which the servos cannot operate, but

identification and simulation of the appropriate failure model is complex. Limitation of the servo speed is however simpler, as we can limit the movement to a maximum speed and thus only meeting a fraction of the demand, should this exceed what the servo is capable of.

One of the key measures of an optimal gait is its stability. Kim et al. proposed a metric for the analysis of gait stability by analysing the centroid of polygons formed by the feet in contact with the ground [63]. This could be used as part of a fitness function which balances gait speed with its stability.

II.3.11 Conclusion

In this Chapter a physically accurate and appropriate, computationally minimalistic model has been developed for the simulation of initially a single leg and then further to a 3D quadruped robot with full dynamics. Expressions have been derived for simultaneous multi-point collision impulse responses and servo torques required to implement the simulated motion. Sample results have been presented attesting to the validity of the models developed. Methods for checking a gait's validity and the torque requirements placed upon the virtual servos are presented for the twofold reasons of facilitating the realisation of such systems and reducing the computational requirements of the simulation. A somewhat pedestrian, statically stable walking gait has been developed to illustrate the application of the quadruped model to realistic locomotion. A physical quadruped robot was created to verify the simulation results and the trajectory of its gait was measured using a laser tracker. A very high degree of agreement was observed between the simulation and measured results with the trajectory exhibiting very similar features, testifying as to the veracity and utility of the simulation developed.

Chapter II.4

Model Verification using Real Robots

The investigations presented within this Thesis are carried out largely in simulation. This is for a number of reasons: first the dynamics of complex non-linear systems can only really be studied numerically as no exact solutions exist. Especially when we consider networks modelling biological neurons or gene expressions, we are forced into a computational approach. The added benefit of this is the powerful way that we can explore all aspects of the behaviour with a 'perfect' set of instruments. Secondly, when we introduce these networks into to an evolutionary framework we must consider the requirement for evaluating typically large populations over many generations. Whilst it is possible to apply artificial evolution to systems evaluated in hardware this typically requires a significant amount of time, presents challenges in maintaining functionality of real robots over extended durations and applies practical constraints on the possible population size and number of generations. For these reasons, typically the evaluation cycle takes place under simulation. However, we must never lose track of the fact that we are dealing here (largely) with embodied systems. To that end we must consider two factors. First, the simulations we design must accurately

model all relevant aspects of the task in the real world (the 'base set' in the language of Jakobi [60, 59]) and secondly, the results we attain must be transferable to the embodied system for which they were developed. This means that we must not only capture the key dynamics of real systems and use this data to validate our simulations, but also design real time control systems capable of implementing the controllers from the simulations. It is only in the application of this cyclic feedback loop of software / hardware validation that we can ensure the utility and accuracy of our results.

The guiding research questions presented in Part I propose the application of the techniques developed throughout Part II to commonly studied tasks in ER. As discussed previously, research in ER often considers both the evolution of adaptive behaviour and robot locomotion and occasionally in the same work. The objective of this Thesis is to investigate the application and efficacy of adding transport delays into RNN connections. Without considering both adaptive behaviour and robot locomotion, hence both wheeled and legged robots, we would be unable to evaluate the efficacy of such as scheme across the gamut of ER research. In this way the work in this Chapter underpins all of the experiments in Part III. Specifically, Section II.4.1 below supports the development of the minimalistic quadruped simulation in Chapter II.3 and the experiment undertaken in Section III.3.2, whilst Section II.4.2 develops the Two-Dimensional (2D) wheeled robot models used in Chapters II.5 and III.2.

II.4.1 Design of a Simple Quadruped

Chapter II.3 develops a computationally minimal 3D dynamic simulation for a legged robot for use in Section III.3.2. In order to determine the accuracy and realism of this model it is necessary to verify its results physically. To that end, this Section describes the design and fabrication of a simple quadruped robot for the explicit intent of comparing the predictions of the

simulation tool with physically observed results.

Legged robots have been investigated for some decades and the purpose of this activity is not attempting to match the state-of-the-art in the mechanical design of locomotory systems (e.g. passive walkers, Robot Cheetah), but rather to provide a simple tool for the investigation of gaits which can easily be included in any study. To this end, the design philosophy for all aspects of the robot, from mechanical to control, is based around low-cost, easily obtainable and open-source products. In this way, anyone can easily recreate these results or use them in a different way, without the barrier of high capital expense as is the case with many commercial legged robot kits. Also, for scenarios where multiple robots are required (for swarm systems or evolution in hardware) minimising the total cost of each individual is of great value. Other researchers have also designed open-source robots for these reasons, for example the Quadratot[1] [125], but the design developed here was cheaper and simpler still.

The scope for this design was therefore to meet the requirements of the simulated gait only and not to exceed them. This leads us to chose the simplest possible combination of materials and parts capable of achieving this.

II.4.1.1 Mechanical Design

A simple construction of foam cored plastic was chosen as it combined light weight, rigidity and sufficient strength to cope with the stresses of the simulated gait. An I-beam cross section was adopted throughout the design with reinforcing braces to ensure a rigid structure. The practicalities of mechanical design over the idealised structure modelled in Chapter II.3 (See Figure II.3.3) require some relaxation of the assumptions made. The most significant of which is that the hip joint requires some offset between the

[1] See, creativemachines.cornell.edu/evolved-quadruped-gaits/.

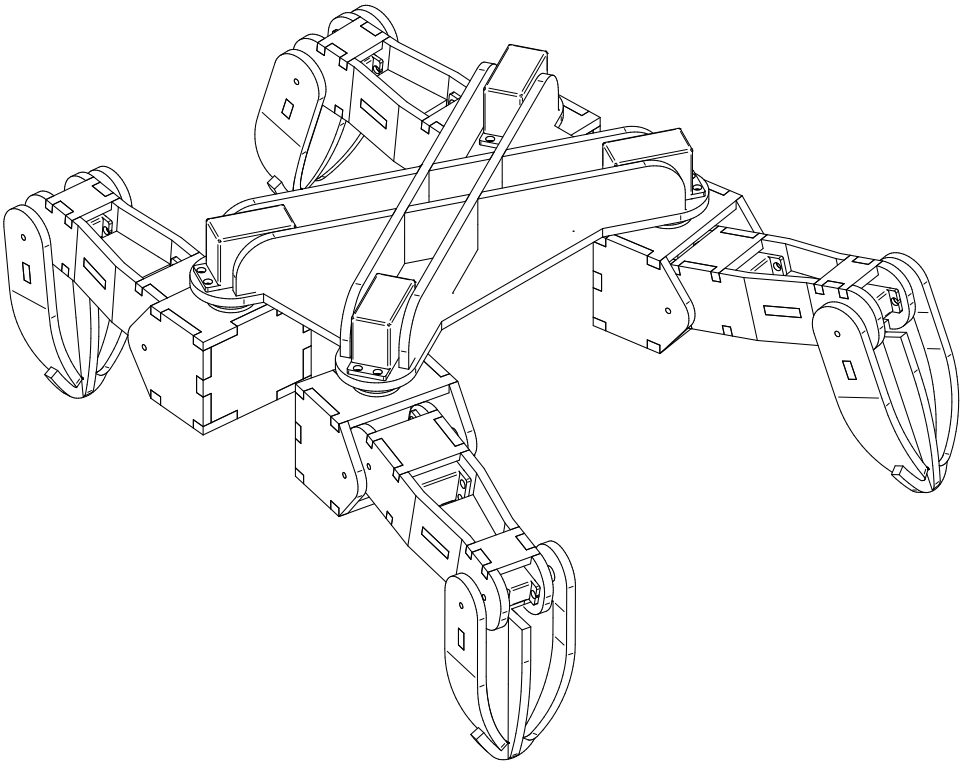


Figure II.4.1: CAD Model of the Quadruped Robot

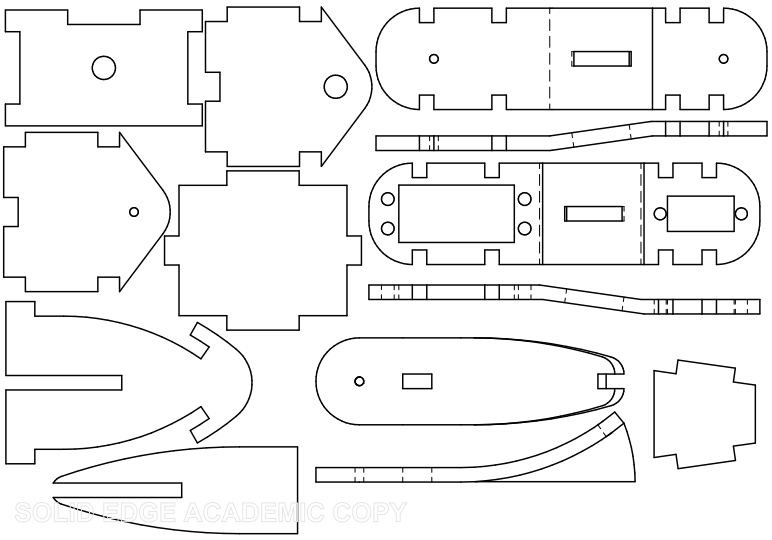


Figure II.4.2: Cut Out Templates for CAD Model

z' and x'_h axes shown in Figure II.3.4. This causes a slight difference in the foot trajectory than that described in Equations II.3.1, II.3.2 and II.3.3. However, the offset is small and the potential differences in feet position and joint torques are negligible (if anything, the n'_β torque would be slightly smaller than predicted as the moment arm is reduced) and outweighed by the desire for simplicity in the modelling process. However, the realistic angle limits specified in Section II.3.1 are feasible.

The mechanical design for the robot is shown in Figure II.4.1 and was designed to be formed from flat sheets of the foam cored material without too much curvature. The structure was designed to match the dimensions of the model skeleton shown in Figure II.3.3 with the dimensions for body size and limb length matching those of the simulation parameters in Table II.3.1. A simple template was generated for each unique part which were typically repeated multiple times, shown in Figure II.4.2.

II.4.1.2 Electromechanical Systems

To actuate the limbs of the robot, we require a method for developing rotational motion. This is most commonly achieved through the use of servo motors. They use DC electric motors in combination with a geared control system in order to provide a limited range of rotational motion. Typically the desired angle is expressed using a PWM signal or digital interface. A servo is specified with the maximum torque it is capable of expressing a speed of response. Most hobby servos are limited to a 180° of motion and are capable of a wide range of speeds and torques. They can range from a couple of pounds to hundreds per servo.

The simulation predicts that the peak torques experienced during the gait for each leg angle will be $|n'_\alpha| \leq 0.05$ Nm, $|n'_\beta| \leq 0.3$ Nm and $|n'_\gamma| \leq 0.1$ Nm. To meet this requirement, the robot was designed to use standard servos for both axes of the hip joints and micro servos for the knee joint.

The maximum torque of each servo (just) exceeds the simulated peak values. Whilst a slightly higher margin would be useful in more practical scenarios, if these servos are capable of powering the gait, we can be sure that the simulation results are accurate (or at least providing an overestimate).

In order to ensure accessibility, an open-source Arduino[2] based controller (DFRobot Romeo V1) was used as an on-board controller. It was ideal for this application as servo control is extremely simple and it is easily employed as an autonomous controller or can be slaved to another device with communication over a wired/wireless serial link.

Figure II.4.3 shows a schematic diagram of the quadruped robot, illustrating the connection of the 12 servos required to drive all four sets of α, β and γ leg angles. Power is supplied by a 7.2V 1800mAh Nickel/Cadmium battery which was regulated using a BEC. This transformed the battery voltage to a regulated 6V with a operational current limit of 3A and maximum of 5A. This corresponds to the maximum voltage of the servos, thus generating the maximum possible torque allowed, only limited by the current draw from the BEC. The Arduino board was powered from the 6V servo power supply.

II.4.1.3 Assembly

Figure II.4.4 shows key steps during the assembly of the robot. First, the templates shown in Figure II.4.2 were traced onto the foam cored composite and cut out. They can then be glued together in sequence, incorporating the servos at the correct moment. Each limb was mounted off servo horns and arranged so that the servos effected motion in line with the feet coordinate systems shown in Figure II.3.4.

The robot was initially suspended under a gantry for testing. This enabled the calibration of leg angles from the installed servo zero positions.

[2] www.arduino.cc

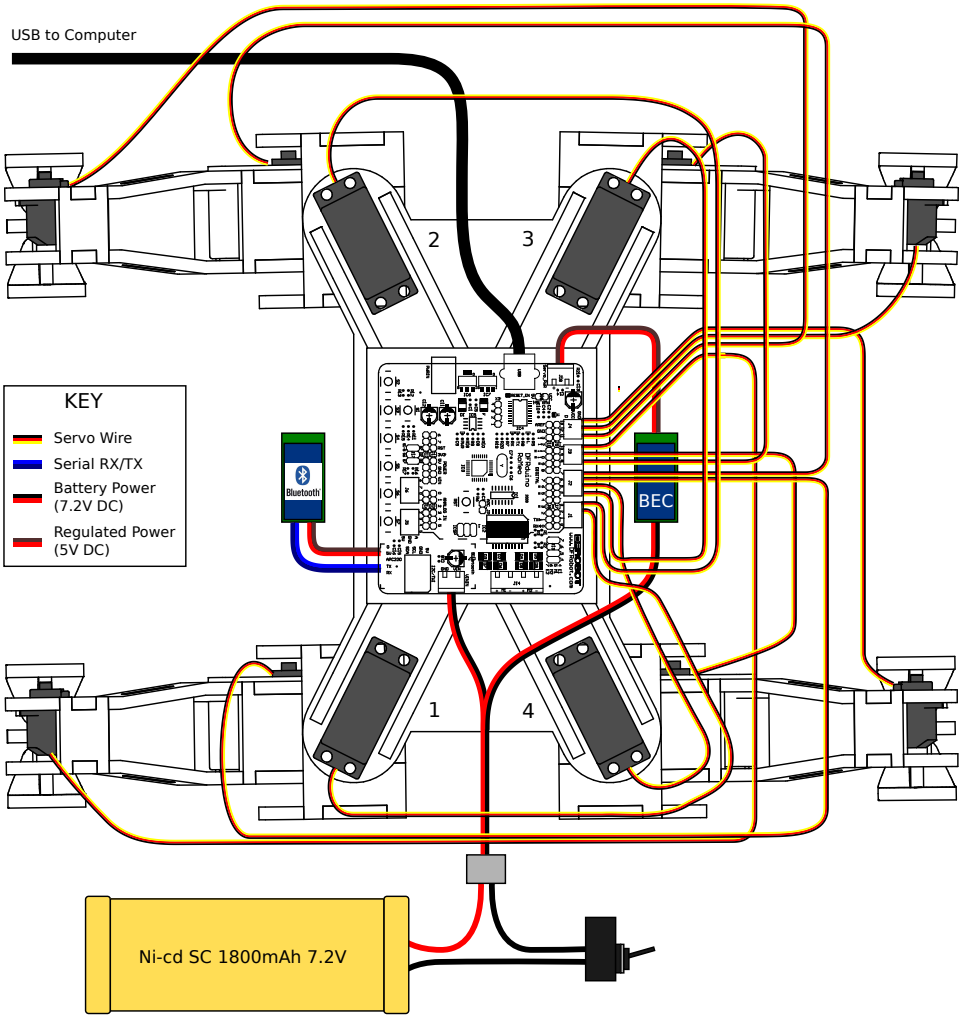


Figure II.4.3: Schematic of Quadruped Robot

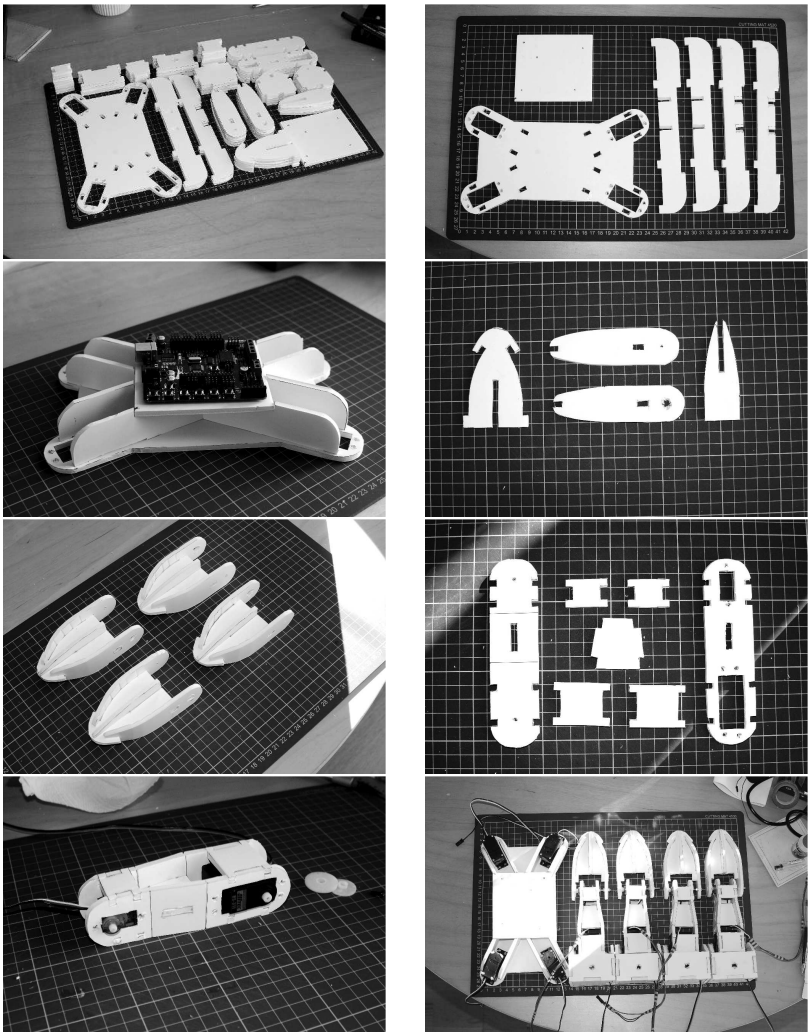


Figure II.4.4: Assembly Steps

The measured offsets were built into the control software embedded on the Arduino board. Once the designed gait had been satisfactorily tested, the robot could then be lowered to the ground and evaluated under gravity. A platform was created for mounting a 1.5" spherical tracking reflector on top of the robot above its centroid. This enabled accurate tracking of its movements using a Leica LTD 500 laser tracker, the results of which can be seen in Chapter II.3 for the designed gait.

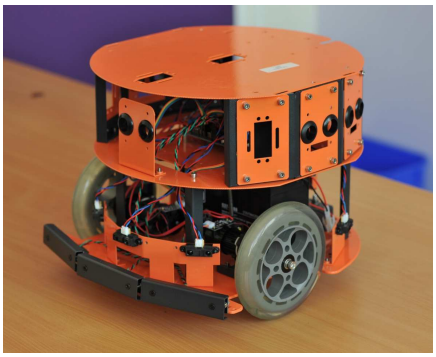
II.4.2 Modelling Wheeled Robots

The recent research focus on legged robots not withstanding, the vast majority of mobile robots for research, industry or the home are wheeled. They tend to only have two wheels, with four (or more) wheeled robots being more generally applicable for outdoors or difficult terrains. There are a number of reasons for this, but mostly this is due to simplicity of design and control. Having two wheels (and some other sliding point or non-driven pivoting wheel) allows a robot to have a zero radius turning circle with precise (assuming appropriate wheel encoders), accurate and easily predicable motion.

As mentioned, this approach is only usually suitable for (relatively) smooth surfaces which usually limits their applications to indoor environments. Exceptions to this certainly exist, the counterweight balanced Segway personal transport for one, but most robots used in general research have two wheels. There exists a phylogeny of research robots of this form, multiple generations of robots developed by K-Team of EPFL in Switzerland, and others.

Figure II.4.5 presents photographs of two of the wheeled robots within the School of Engineering and Computing Sciences, namely the Khepera III robot by K-Team of EPFL, Switzerland and the DFRobot HCR Platform. They are very similar in terms of their form factor, varying only in terms

(a) K-Team Khepera III



(b) DFRobot HCR Platform

Figure II.4.5: Real Wheeled Robots

of scale and the sophistication of on-board systems, with: two DC motors, wheel encoders, a battery, on-board control system and a range of sensors distributed around the robot.

The requirements to model the motion and environmental interactions of these robots in simulation has been previously discussed. We could employ a real-time evolution in hardware scheme, but the constraints to any investigations applied render the discovery of non-trivial solutions unlikely. To this end then, we must be able to accurately simulate their behaviour in a physically plausible manner, but with the minimum computational cost possible to reduce the duration of extended evolutionary runs.

II.4.2.1 Geometrical Motion of a Two-wheeled Robot

The modelling of two wheeled robots typically presents a significantly decreased challenge as we can often represent them purely in two dimensions, ignoring three dimensional issues, thus simplifying the computational requirements of any simulation. By assuming the wheels of the robot do not slip (except in some instances where we may model collision with walls by slippage) and ignoring inertial effects we can simply consider the path described by the robot as a geometrical relationship determined by the relative

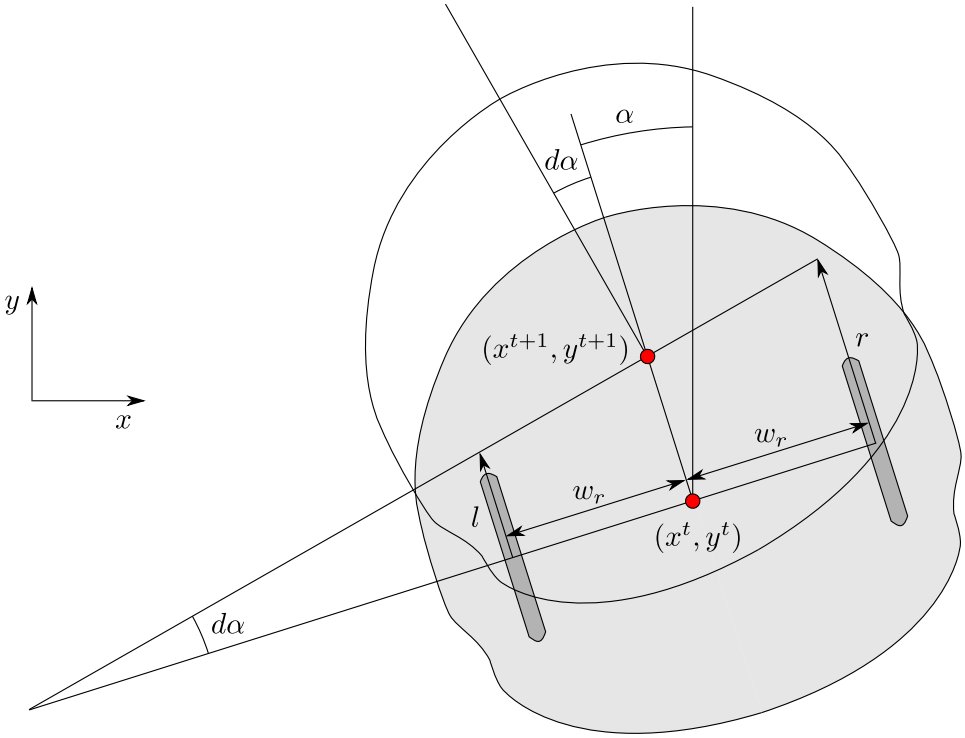


Figure II.4.6: Two Wheeled Differential Drive

driving motions of each wheel.

Jakobi [59, 60] adopted a lookup table approach for a minimal simulation of two-wheeled robots. However, this was limited to the corridor environment for the task under investigation, and not generally applicable to general motion in an arbitrary environment.

Figure II.4.6 shows a simple geometrical approach to two wheeled robot motion. Here, the trajectory between time steps is approximated as a straight line segment, with the angular deviation in the robot's heading derived from the differential drive of the left and right hand motors.

The Khephera robot takes a motor set speed in the range of -43,000 to 43,000 (numerical values which relate to the internal encoders), which can be converted to millimetres per second by dividing by 144.010. In this way, neural activations (0 to 1) can be directly converted into the distance

travelled by each wheel contact point during a time step.

Following the conventions shown in Figure II.4.6 the following identities can be easily derived:

$$x^{t+1} = x^t + \frac{l+r}{2} dt \sin\alpha \quad \text{(II.4.1)}$$

$$y^{t+1} = y^t + \frac{l+r}{2} dt \cos\alpha \quad \text{(II.4.2)}$$

$$d\alpha = \text{atan}\left(\frac{dt(l-r)}{w_r}\right) \quad \text{(II.4.3)}$$

These expressions are extremely simple and carry a very low computational cost, especially if pre-calculated lookup tables are employed to approximate trigonometric functions, and do not present a noticeable increase in computational complexity over the work of Jakobi [59, 60].

Typically we seek to penalise robots who contact the walls in simulation. Often we may wish to allow them to continue driving in contact with a wall, whilst suffering a cumulative fitness penalty. This can be simply modelled by allowing motion away from or parallel to the wall and reducing the movement per time step by a factor to represent the friction of the walls on the robot. This approach ignores the turning moment that the contact will apply, but as this would necessitate the modelling of the frictional contact and slippage of the wheels in a much more thorough manner it may be disadvantageous to do so.

II.4.2.2 Simulation of Distance Sensors

Figure II.4.7 shows a two dimensional representation of a two-wheeled robot, with a position $\mathbf{p} = [p_x, p_y]^T$ and heading α. We can extend a ray from the location of a sensor $\mathbf{d}$ at an angle α_s to that of the robot heading. This ray angle intersects the longitudinal axis of the robot ($\alpha_s = 0$) at point $\mathbf{a}$ which is offset from $\mathbf{p}$ by s_o along the longitudinal axis. The sensor location is a known distance s_d from $\mathbf{a}$ along the ray which allows for irregular shaped robots and sensor arrays. We then assume that the ray intersects a wall

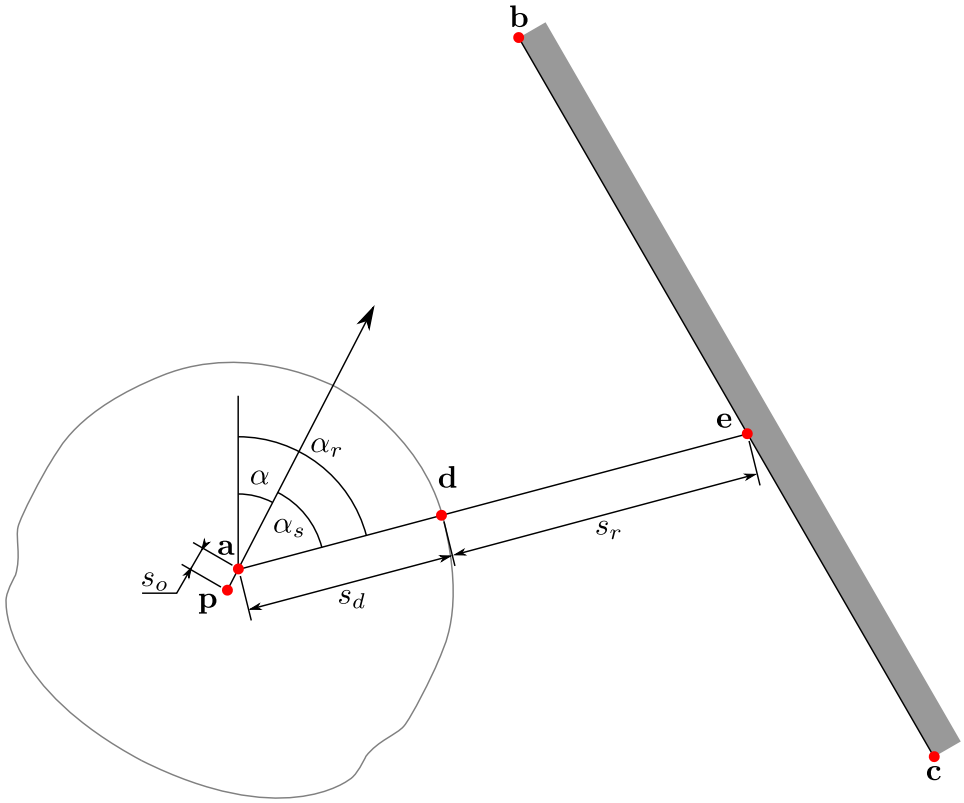


Figure II.4.7: Sensor Distance Geometry

in the environment, defined by its end nodes **b** and **c**, at **e**. The sensor distance which we wish to find is then given by s_r, and can be found using the expressions presented below.

$$\alpha_r = \alpha + \alpha_s \tag{II.4.4}$$

$$a_x = p_x - s_o \sin\alpha \tag{II.4.5}$$

$$a_y = p_y - s_o \cos\alpha \tag{II.4.6}$$

$$d_x = a_x + s_d \sin\alpha \tag{II.4.7}$$

$$d_y = a_y + s_d \sin\alpha \tag{II.4.8}$$

$$f = \frac{d_x - a_x}{d_y - a_y} \tag{II.4.9}$$

If $(c_x - b_x) - f(c_y - b_y) \neq 0$, which checks that the lines do intersect at some point (i.e. not parallel):

$$h = \frac{(a_x - b_x) + f(b_y - a_y)}{(c_x - b_x) - f(c_y - b_y)} \tag{II.4.10}$$

If $h > 0$ and $h < 1$, which tests for whether the point intersection lies between the end points of the wall:

$$v = \frac{(b_y - a_y) + h(c_y - b_y)}{d_y - a_y} \tag{II.4.11}$$

If $v > 0$, which checks if the intersection point is in the positive direction $\mathbf{d} - \mathbf{a}$ and not in the reverse direction $\mathbf{a} - \mathbf{d}$:

$$\mathbf{e} = \mathbf{b} + h(\mathbf{c} - \mathbf{b}) \tag{II.4.12}$$

$$s_r = |\mathbf{e} - \mathbf{d}| \tag{II.4.13}$$

$$s_r = \sqrt{(e_x - d_x)^2 + (e_y - d_y)^2} \tag{II.4.14}$$

The expressions derived above will give a distance directly. However, we must also consider a reverse mapping from the true distance to a modelled sensor output for inclusion within the simulation. The robots shown in Figure II.4.5 are equipped with IR proximity sensors and Ultrasonic distance sensors for environmental feedback. We must therefore characterise

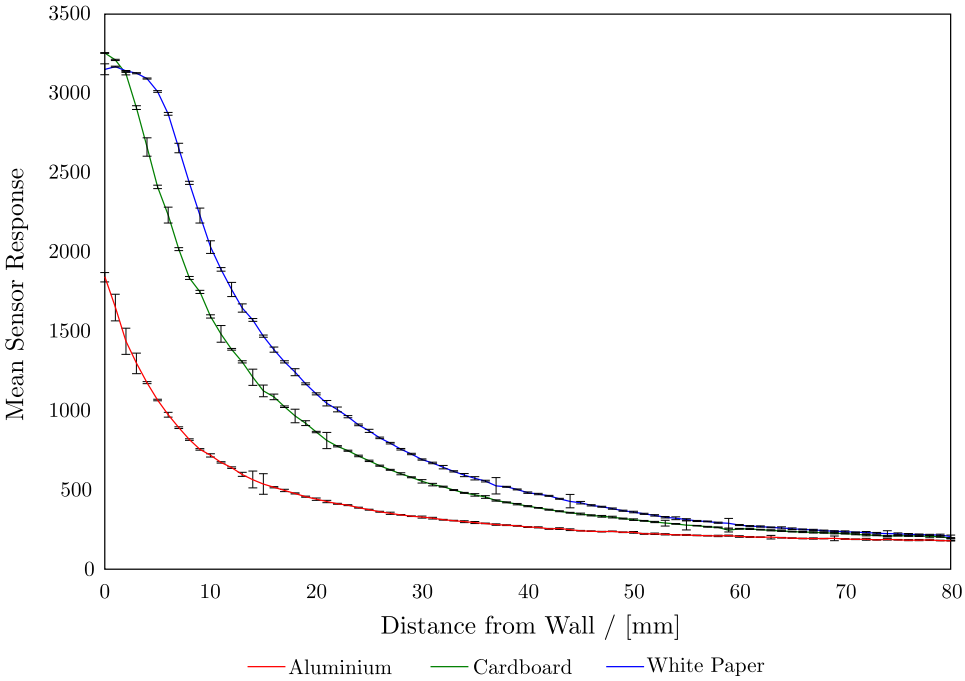


Figure II.4.8: IR Sensor Calibration

the sensors on the real robot to ensure the accuracy of the simulation and ensuring the applicability of evolved controllers. A Khephera III robot was programmed to record a large number of sensor samples at a range of distances to determine the response curve and the uncertainty associated across the range of the sensor (around 80mm). These curves are shown in Figure II.4.8 for three different materials commonly encountered in research test environments. Using lookup tables with simple linear interpolation we can easily model this non-linear mapping, with measured levels of random noise added for realism[3] [60, 59].

[3]In fact, Jakobi demonstrated how CTRNNs use the noise in the system to develop better solutions.

II.4.3 Common Code for Simulation and Verification

As shall be discussed in Chapters II.5 and II.6, the evolutionary framework used in the investigations presented throughout this Thesis is coded in the Python programming language.[4] This may seem a self-contradictory approach for implementing simulations which by their very nature require many runs and typically are focussed on maximum run speed and minimal computational complexity. Putting this aside for a moment, there are some advantages in using Python. The language is very high level, interpretive, platform independent and open source with a vast array of community maintained highly efficient modules for almost anything. This enables us to run the same code used in simulation to control robots in real time on embedded hardware or over wireless or serial links. Not having to compile code and access to powerful plotting libraries allows for accelerated development and prototyping of programs. However, this comes at the expense of slower run time performance compared to compiled languages due to the Just-In-Time (JIT) compiler.

This would seem to render Python unsuitable for ER and related disciplines, but it is possible to accelerate Python code so that it approaches the speed of native C. The PyPy[5] is an implementation of Python 2.7.1 in Python itself. It uses a restricted subset of the language (RPy) with a limited support for Python modules. With these compromises, PyPy can translate and compile Python into a variety of target platform independent languages, e.g. C and Java. This is not only true for the translated Python JIT complier, but also for specific standalone programs. The degree to which any program can be accelerated is highly variable, but the geometric average of all benchmarks run by the project is 5.7x faster than CPython.[6]

[4] http://python.org.

[5] http://pypy.org.

[6] http://speed.pypy.org - Accessed on 30th November 2012.

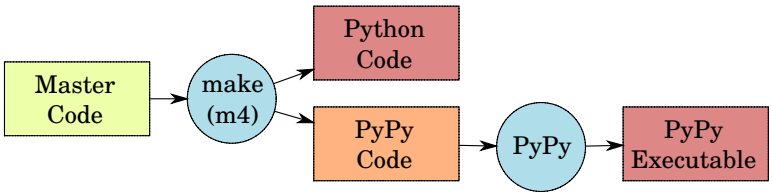


Figure II.4.9: Common Code Pipeline Schematic

This capability allows a combined approach between the ease of development, visualisation and analysis of a high level interpretive language with the accelerated performance required for long, computationally intensive simulation runs, shown schematically in Figure II.4.9. The key to this approach is the maintenance of a Master code library which is independent of the target, be it Python or PyPy. The restrictions placed on PyPy require the Master code to include implementation specific code snippets which are selectively removed when Make (using m4, which is used to remove all lines in a file which are marked with a specific tag) is run to generate the target Python and PyPy code. This means that simulation code cannot have embedded plotting, but rather prints results to a file for later analysis in traditional manner. As long as the analysis script does not contain any removal tags it will remain unchanged during the translation and can be run in parallel with the simulation, for example analysing the results of each generation as it is completed as well as generating summary plots. The makefile used for this process is shown in Listing II.4.1 for completeness.

The makefile identifies all Python files under the Master folder, up to two levels deep, as sources. It then copies all these files to target directories for the Python and PyPy specific translations, using the m4 rules shown in Listings II.4.2 and II.4.3 to remove incompatible tagged code lines after which each file is then made executable. After which, the PyPy code can be compiled using the PyPy translation script which produces the final executable program. Once simulation results have been collected, it is then

Listing II.4.1: Makefile Code

```
SOURCES = $(wildcard Master/*.py) $(wildcard Master/*/*.py)
    $(wildcard Master/*/*/*.py)
PYPY_TARGETS = $(subst Master,PyPy,$(SOURCES))
PYTHON_TARGETS = $(subst Master,Python,$(SOURCES))

all: copy $(PYTHON_TARGETS) $(PYPY_TARGETS)

copy:
    @mkdir -p PyPy/
    @mkdir -p Python/
    @sudo cp -r Master/* PyPy/
    @sudo cp -r Master/* Python/
    @sudo find . -type d -exec chmod 775 {} \;
    @sudo find . -type f -exec chmod 664 {} \;
    @sudo find . -type d -exec chown Username:Username {} \;
    @sudo find . -type f -exec chown Username:Username {} \;

$(PYTHON_TARGETS): copy
    cat m4_python $@ > /tmp/out && sudo mv /tmp/out $@
    m4 -P $@ > /tmp/trans && sudo mv /tmp/trans $@

$(PYPY_TARGETS): copy
    cat m4_pypy $@ > /tmp/out && sudo mv /tmp/out $@
    m4 -P $@ > /tmp/trans && sudo mv /tmp/trans $@
```

Listing II.4.2: m4 Python Rules

```
m4_changequote(¬,¬)m4_dnl
m4_define(Python,$1)m4_dnl
m4_define(PyPy,¬m4_dnl¬)m4_dnl
```

Listing II.4.3: m4 PyPy Rules

```
m4_changequote(¬,¬)m4_dnl
m4_define(PyPy,$1)m4_dnl
m4_define(Python,¬m4_dnl¬)m4_dnl
```

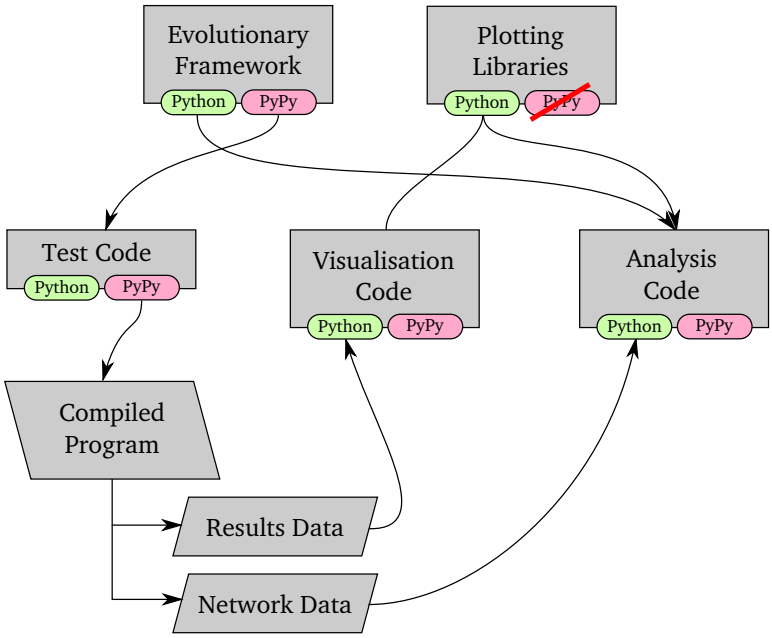


Figure II.4.10: A Common Code Approach

simple to import evolved networks into Python code, referencing the core modules of the evolutionary framework, to visualise, interrogate and experiment with results using the exact same code as drove the simulation itself. This is one of the key advantages of this approach, being that we can have a great deal of confidence that the behaviour of simulations or network evaluations will be identical between Python and PyPy versions of the software. Figure II.4.10 shows a general example of how this common code approach is used in the investigations presented in this Thesis.

II.4.4 Real Time Control Architecture

The overarching philosophy utilised throughout employs open-source, platform independent approaches, showcased in Section II.4.3. In line with this, we wish to control a range of robots using the same core modules for network evaluation as used throughout the simulation and analysis of evolved solutions. A common communications protocol is therefore required to allow a common control approach across multiple platforms (with different sensors, motors, control systems, etc.). Figure II.4.11 shows a schematic of the real time control architecture employed throughout the work presented in this Thesis. Here, a host PC uses the Python version of the evolutionary framework (see Section II.4.3) to evaluate the evolved network solutions, based on real time inputs. A call/response serial communications protocol allows the control program to control the robots and request any sensor data available for input into the network under evaluation. This can be achieved using both wired and wireless systems depending on the particular use case for the robot, but crucially the messages transmitted are to be independent of the transport medium. This additional layer in the control of the robot, as compared to an entirely embedded control system, includes a lag term which could potentially affect the behaviour of the system. However, this is small and can easily be modelled for inclusion into the simulations if required.

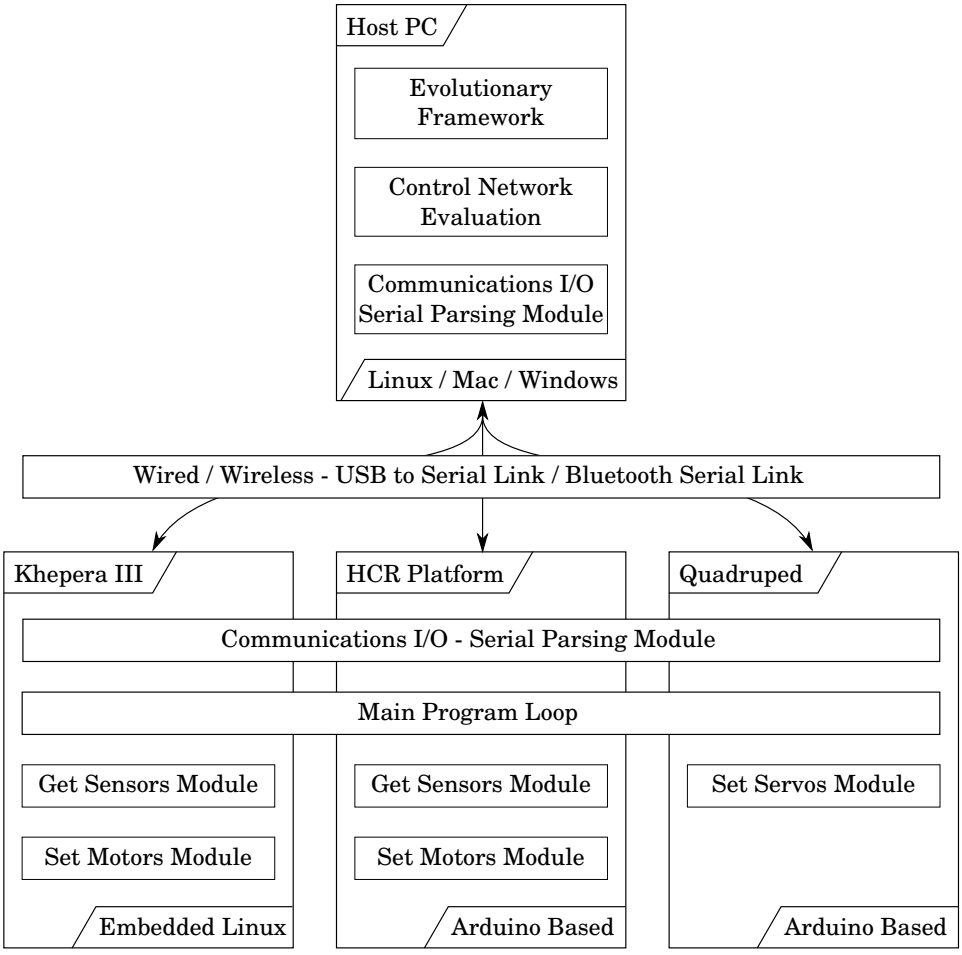


Figure II.4.11: Real Time Control Architecture

The heterogeneous nature of the robots means that there can only be so much common code between the slave programs embedded on each. As the control system may be very different, varying from embedded Linux running C, Python or any other suitable language, to Arduino systems using their own language based on Processing.

II.4.5 Conclusions

In this Chapter, we have discussed the importance of closing the loop between evolving controllers in simulation and the evaluation of said controllers on real robots. By allowing our knowledge of the real robot to inform the design of the simulation we render the development of controllers capable of working in the real world, which after all is the focus of ER.

The design, manufacture and assembly of a quadruped robot for the sole purpose of verifying the simulation developed in Chapter II.3 is presented. The robot used low cost servos, an open source Arduino controller and simple foam cored construction to minimise costs and the barrier for a third party to recreate all or part of the related work presented.

Along with the quadruped robot used in Chapter II.3, the examples presented in Chapters II.5 and III.1 use a range of two wheeled robots. Simple two-dimensional models for simulating robot motion and sensor readings are presented alongside real data used to calibrate the non-linear IR sensor mapping.

It is important to have confidence in the equivalence of behaviour for code running across different platforms and devices. A common code approach has been developed where a master copy of the evolutionary framework and test code is compiled into pure CPython and PyPy accelerated executables for extended runs. This harnesses the power and rapid prototyping of Python with speed of execution comparable with a non-interpretative programming language, giving the researcher the best of both worlds. In this

way, the same code is used in the development, analysis and verification of evolved solutions.

This approach is carried right through to the real time control of robots using evolved networks using a host / slave approach. The host PC runs the CPython framework and governs the behaviour of a range of robots using a common two-way platform and medium independent communications protocol.

In this way, we can guarantee the compatibility and equivalence of network behaviour through development, analysis and real time evaluation of robots. This fundamentally sound basis not only serves to support the validity of later results presented but also to accelerate the development and testing of examples in practice.

Chapter II.5

Evolving Adaptive Behaviour with Time Delays

For the neuroevolution of robotic controllers capable of learning and adaptive behaviour it has been argued that we must simply allow artificial evolution to exploit dynamic neural behaviour over a range of time scales. CTRNNs go a long way towards increasing the complexity of network dynamics sufficiently to enable the emergence of this type of desired behaviour as discussed in Chapter II.1. In biology, however, the time for a neural signal to propagate through the brain or delayed regulatory effects in GRNs is thought to be vital in contributing to the dynamics of the system. In this Chapter the extended neuron model developed in Chapter II.2 is used to simply model these delays and applied for the first time, to the author's knowledge, to a robotic learning task. The evolved networks are then analysed to prove the necessity of synaptic time delays for the correct behaviour of these systems. By demonstrating the practical impact of the behaviour differences discovered in Chapter II.2 in the context of an adaptive behaviour ER task, this Chapter goes some way to answering the research questions introduced in Part I and supporting the experimentation of Part III; indeed the T-task task used here is expanded upon in Section III.2.1

II.5.1 Evolutionary Architecture

To demonstrate the application of the extended neuron model developed in Chapter II.2 to adaptive behaviour, it is necessary to select an evolutionary architecture in which to develop examples. Stanley developed a novel open ended evolutionary framework where minimal genotypes evolve through complexification in a method called NEAT [104]. It has been demonstrated to be more efficient than traditional methods and has great potential for the discovery of complex solutions whilst spending the majority of the evolutionary process in low-dimensional search, building the foundations of good solutions. This process of complexification is biologically plausible, for there is good evidence that throughout the evolution of the brain small numbers of neurons were repeatedly added to the existing structures [76].

A Python implementation of the NEAT Methodology of Stanley and Miikkulainen, written by Brian Greer,[1] was modified for the purposes of demonstrating the application of the CDRNN model. The default implementation of the NEAT method in pyNEAT is to use a simple discrete neuron model of the form $y_i = \sigma(\sum_{i=1}^{N} w_{ij} y_j)$, where $\sigma = 1/(1 + \exp^{-x})$. This was modified to incorporate the neuron model introduced earlier. The neuronal time constants (T_i) are initialised with a common value and with a given probability perturbed during mutation operations, as are the weights of synaptic connections.

This Python implementation allows for a powerful degree of interactivity in development and greatly accelerates prototyping of simulations, but its interpretative nature renders it unsuitable for direct application to highly computationally intensive evolutionary runs. PyPy[2] was used to translate and compile finished simulations into C executables which approach the

[1] pyNEAT Version 0.0.2, http://code.google.com/p/pyneat/, released under GNU General Public License, version 2.

[2] PyPy, http://pypy.org/.

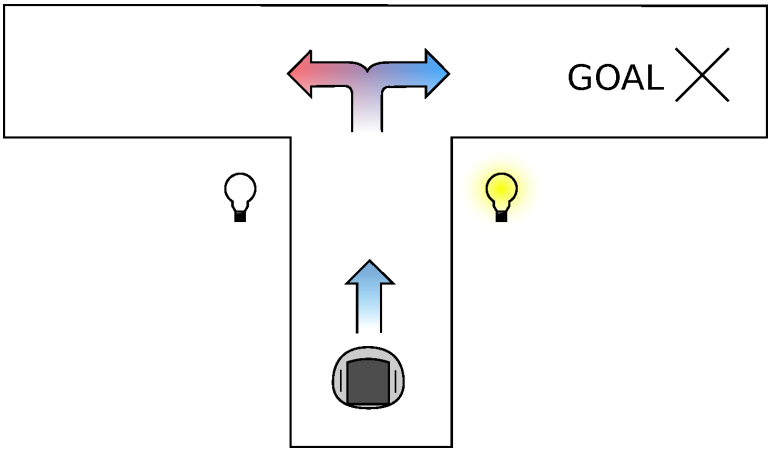
execution speed of native C code, thus rendering the whole exercise feasible. In this way long runs which would take weeks execute in days.

II.5.2 T-Junction Task

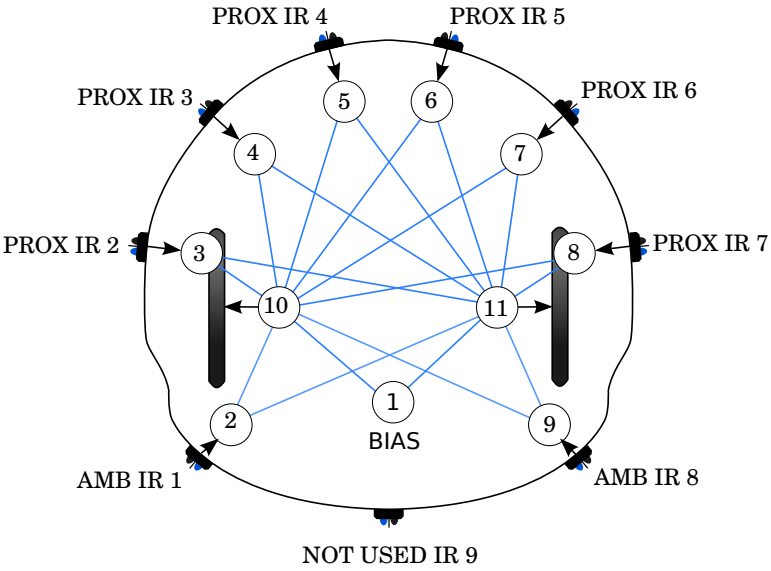
The example chosen to demonstrate this system is that of a T-Junction test, a task well studied in ER [59]. It is this fact which makes it so suitable as it establishes the capability of the system under study, before moving on to novel, and more challenging, problems. It is simple, but requires behaviour beyond that of the simply reactive and necessitates a good solution to develop a behavioural switch by associating a light activation with a directional choice at a T-Junction. A cartoon of the task is shown in Figure II.5.1a.

The robot chosen for use in these simulations was modelled on the Khepera III research robot by K-Team of Switzerland, as these robots have been used a great deal in ER literature and were used for this task by Jakobi [59]. The real robots are available for experimentation and generation of accurate sensorimotor simulations for the evolutionary run, and the results of the computer simulations may then be tested in the real world (see Chapter II.4). The Khepera robot has two wheels and a variety of IR and Ultrasound (US) sensors around its quasi-cylindrical form. A top view schematic of the robot, illustrating a minimal genotype for the NEAT methodology, is shown in Figure II.5.1b.

The robot is placed at the end of a corridor with a randomised starting angle ($-\frac{\pi}{6} \leq \alpha \leq -\frac{\pi}{6}$) and x-offset from the corridor centreline ($-50mm \leq x_{offset} \leq 50mm$). The T-junction geometry may be seen in Figure II.5.4b and consisted of a straight corridor centreline of length 1000mm intersecting another corridor centreline of length 1000mm perpendicularly at its centre-point. The walls lie 125mm either side of these centrelines forming a corridor 250mm across. The sensor inputs are calculated at each time step by tracing



(a) Cartoon of T-Junction Task



(b) Khepera III Top View

Figure II.5.1: T-Junction Task & Network

a ray cast from each sensor to the closest intersecting wall, looking up the appropriate sensor response from measured physical data, with the addition of a physically plausible level of random noise ($\pm 2.5\%$). These values were scaled between zero and one, where one would correspond to the maximum possible sensor reading, and input to the network. The outputs of each node were evaluated using an Euler numerical integration of the governing differential equations. The motor outputs (m_l,m_r) were calculated from the two output neurons (o_1, o_2) using the following equations:

$$m_l = o_1(1 - o_2) \tag{II.5.1}$$

$$m_r = o_1 o_2 \tag{II.5.2}$$

As neuron outputs have a possible range of zero to one, the first motor output governs the speed and the second turns the robot. The calculated motor demands were scaled to between zero and the maximum motor speed. Once the robot has successfully navigated along the corridor to a specific point a light is illuminated on one side of the robot. This is simulated by exciting the appropriate ambient light IR sensor with a measured high or low value. The robot then approaches a T-junction at which it should turn in the direction indicated by the light. If it reaches a goal area at the end of the corridor, it is rewarded with a significant boost to its fitness. The overall fitness function used in these experiments is dependent upon the robots horizontal and vertical positions (in mm), the direction the robot turns, and whether it reaches the goal area:

$$f = \begin{cases} m \times \text{abs}(x) + y + r & : \text{if turned correctly} \\ -\text{abs}(x) + y + r & : \text{otherwise} \end{cases} \tag{II.5.3}$$

Where r is calculated as follows should the robot reach the goal area. If it

does not, the reward is zero:

$$r = r_{max} * \frac{(T-t)}{T} \tag{II.5.4}$$

$$r_{max} = \begin{cases} 5000.0 & p = 0 \\ 2500.0 & p \neq 0 \end{cases} \tag{II.5.5}$$

The robot is permitted to contact the walls of the environment, but a penalty term $p = 5$ is subtracted from its total fitness for each timestep that it does so. Therefore a p value of zero indicates a clear run and corresponds to a larger fitness bonus on reaching the goal. The goal area, shown filled in Figure II.5.4b, prevented the end of the corridor being reached. Thus, the maximum fitness attainable by a robot for each trial was 8500.0 and the lowest was limited to 0.0. the fitness function has been deliberately designed so that it is impossible to achieve a perfect score given a finite movement speed and to show a clear decrease in fitness should the robot take longer to reach the goal, or if it touches the walls of the environment. In this way there was an evolutionary pressure to develop more efficacious controllers whilst ensuring that initial generations could still perform sufficiently to avoid the Bootstrap Problem.

The evolutionary run was initiated with a minimal genotype of nine input nodes (one bias node with a constant value of 1.0 and eight IR sensors as shown in Figure II.5.1b) and two output nodes which control the robot motors as previously described. The seeded default network parameters (τ_i = 1.0, w_{ij} = 0.0,θ = 0.0, and T_{ij} = 1.0) were varied randomly to generate an initial population of unique configurations. Each individual evaluation consisted of four trials: two of each possible configurations of lights and goal zones. Each trial started with unique randomly generated positions and headings for the robot to instil generality of behaviour in the evolved solutions. The evolutionary simulation ran for 400 generations, the fitness history of which is shown in Figure II.5.2.

A common issue across ER is the trade-off between the detail of an

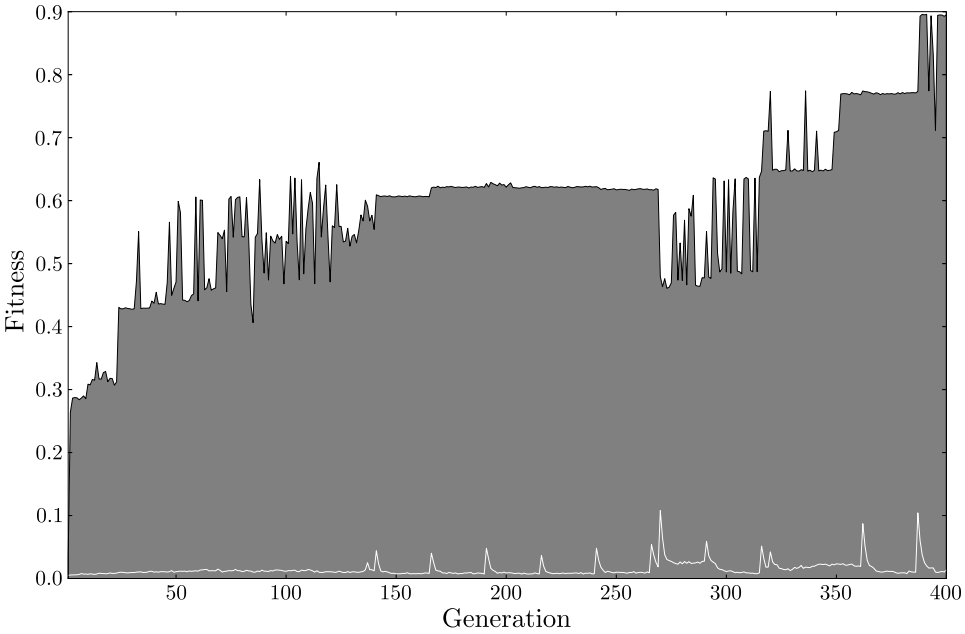


Figure II.5.2: Normalised fitness history; 1000 individuals, 400 generations.

experiment, be it the degree of realism, number of trials, size of population, number of generations and many more parameters besides, and the computational cost to enable feasible experimentation. Typically, this balance is achieved through the experience of the designer and trial and error. Sensitivity analysis can be complex and computationally onerous given the number of parameters which intimately affect the evolutionary process. The four trials per individual evaluation aimed to minimise the computational cost (doubling the trials doubles the execution time) and yet still expose the evolved controllers to variability in the environment to establish generality in the manner of Jakobi [59]. As demonstrated in Section II.5.3, the solutions evolved with four trials per evaluation were found to perform reliably when tested with much higher levels of repetition (100), validating this aspect of the experimental design.

The population went through several stages of capability before arriving at an optimal solution. Initially, the robots would successfully navigate to

one goal whilst sliding along the wall most of the time, resulting in a low fitness ($\approx$ generation 25). By generation 50 the individuals were using the light stimulus to turn appropriately, but unreliably and with a considerable number of collisions. Through successive generations the behaviour was tuned to maintain a constant distance from the walls, turn according to the stimulus, and reach the correct goal the majority of the times. However it was only in the final generations of the run that an optimal behaviour evolved capable of navigating the corridor at maximum speed and reaching the goals, without any wall collisions.

The NEAT parameters ensured an exploratory search, thus a proportion of the individuals in each generation achieved zero fitness by repeated wall collisions. This lead to a relatively low, but stable, average population fitness throughout the run. The best evolved network of the final generation is shown in Figure II.5.3, with trajectories for 100 runs of 4 trials shown in Figure II.5.4b. The input nodes are arranged at the bottom of the figure and the four output nodes are arranged at the top. Some additional recurrent synapses and a hidden node have been added through the complexification process inherent in the NEAT method.

In Figure II.5.3 the time constant of each neuron is proportional to the size of the network node (a small node indicates a small, fast nodal time constant), and the weight of each synaptic connection is indicated by the thickness of the connection (overlapping synapses are shown separated and curved for clarity). A red line indicates a negative (inhibitory) connection and a black line shows a positive (excitatory) connection. The gradient of each connection (dark to light) indicates the direction of signal propagation and each dot indicates one 'store' in the synapse queue; thus a synapse with five stores would take 5 time steps for the signal to propagate through it.

The full genome for the individual is too long to display here (12 neurons and 35 synapses), but the statistical characteristics of the synaptic time

	μ	σ	Max	Min
Synaptic Time Delays (τ)	1.779	2.293	6.432	0.200
Node Time Constants (T)	0.466	0.444	1.891	0.030

Table II.5.1: Genome Statistical Characteristics

delays and node time constants are given in Table II.5.1. The range of values seen attests to the high degree of variation present in the parameters characterising the evolved 'fit' solution.

For both parameters the minimum possible value was $\tau_{min} = 0.05s$ and $T_{min} = 0.2s$ which were enforced as the smallest possible values for the appropriate modelling of the time delay and dynamic response given the integration time step.

The evolutionary process has been shown to successfully solve the task and, whilst it was possible for the genetic operators to almost zero the temporal parameters, they did not and the best solutions contained considerable delays. The important question however is whether the evolutionary architecture achieved this despite the delays, or were they useful in the development of fit solutions. Furthermore what sensitivity is there to the evolved set of delay values for fit behaviours to be maintained?

II.5.3 Sensitivity Analysis

In order to evaluate the degree to which the system has evolved a dependence on the specific configuration of time delays, it is possible investigate the effect of changing them on behaviour and fitness. First, the time delay associated with each synapse in the network was scaled by a range of factors and the network performance evaluated. Each run consisting of four trials was repeated one hundred times with random starting position and sensor noise to smooth out stochastic effects. This degree of repetition was

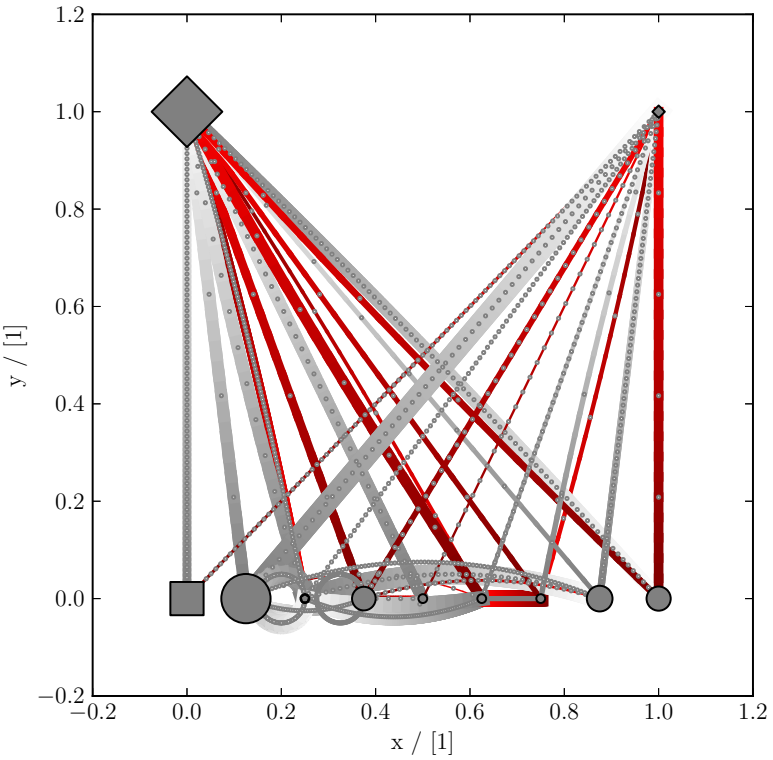


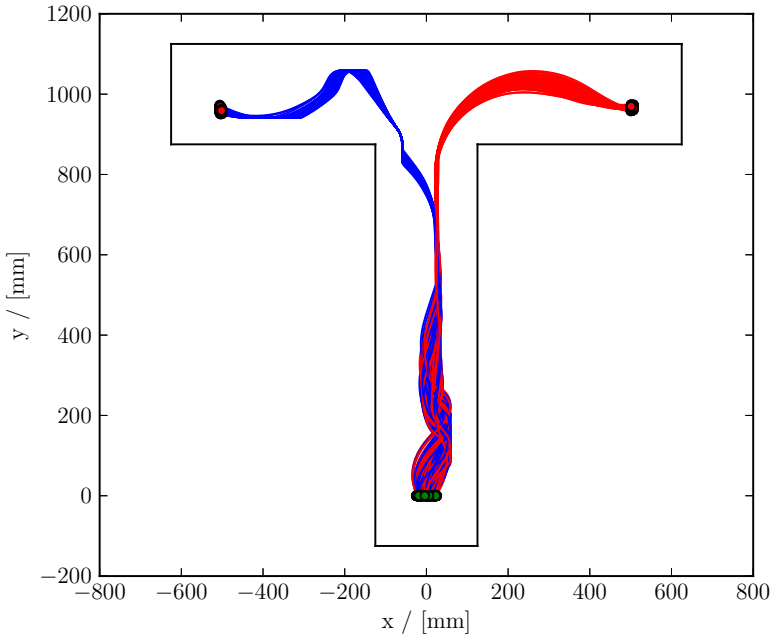
Figure II.5.3: Evolved Network

chosen as a result of repeated runs to establish a practicable and reliable number. This is to a degree arbitrary and is not determined as a result of formal sensitivity analysis. However, this represents a practical computational limit and as can be seen in Figure II.5.4 the range of possible starting headings is well explored. Figure II.5.4a and II.5.4b demonstrate how even with this variability in starting condition reliable results are achieved and stochastic effects are eliminated. The random nature of the results shown in Figure II.5.4c and Figure II.5.4d are solely the result of scaling the time delays. That said, further work may benefit from more formal consideration of sensitivity of these (and other) results to the number of repetitions.

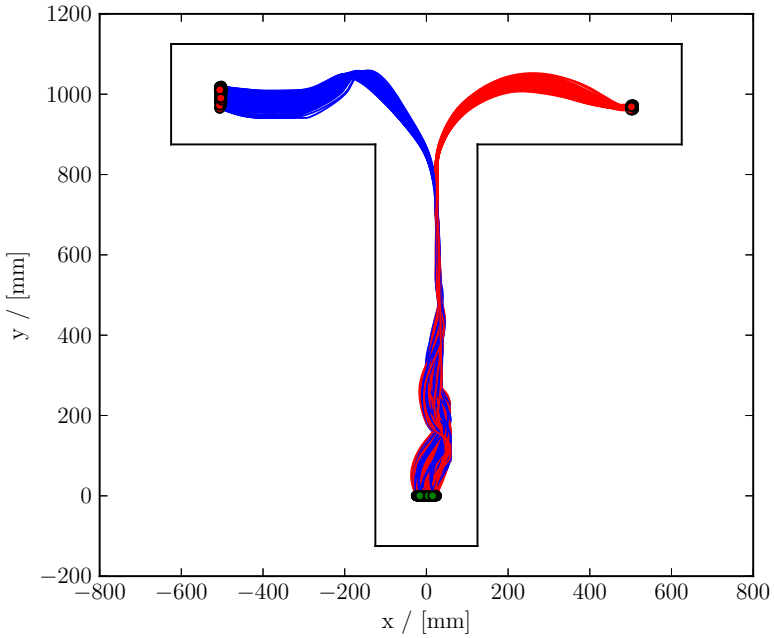
The effect of scaling the synaptic delays on the resulting trajectories is shown in Figure II.5.4. A scale value of zero equates to a CTRNN version of the network; all other parameters of the network other than the time delays are kept constant throughout all of the trials. The fitness values shown are for one trial and consequently have a maximum value of 34,000 (four runs of 8500).

As the time delays are increased the behaviour of the robots quickly become somewhat erratic. Turns are over-accentuated leading to a decreased fitness. The left turn behaviour also begins to fail in a noticeable proportion of the results, with the robot reflecting off the environment boundaries and ending up in the incorrect goal area, whilst the right hand turn remains unaffected (Figure II.5.4c). When the delay scaling gets larger complex rebounding trajectories are present for all trials and a large number of runs fail to reach any goal and reach the end of the evaluation time fairly evenly distributed around the environment (Figure II.5.4d). Extreme values of scaling, six times onwards, result in very few trials even reaching the T-junction and a total breakdown in any form of desirable behaviour.

Like with the example given in Figure II.2.1, the increased delays lead to larger transitory responses which cause a significant alteration to the tra-

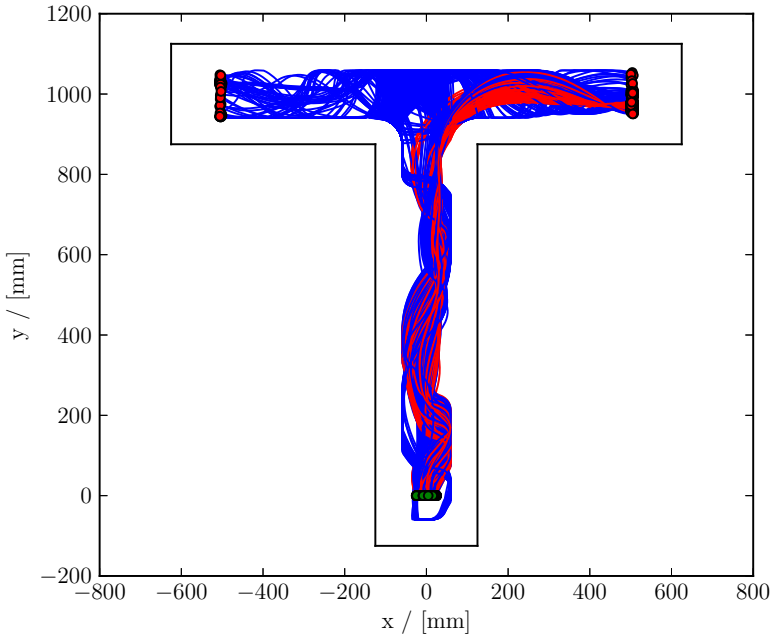


(a) None

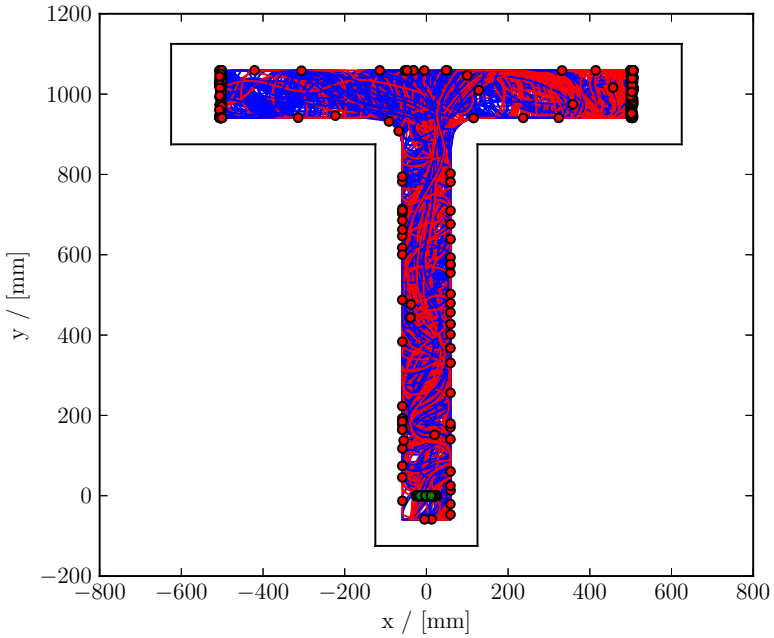


(b) Evolved delays

Figure II.5.4: The effect of scaling synapse time delays on trajectories



(c) Twice



(d) Four times

Figure II.5.4: The effect of scaling synapse time delays on trajectories (continued)

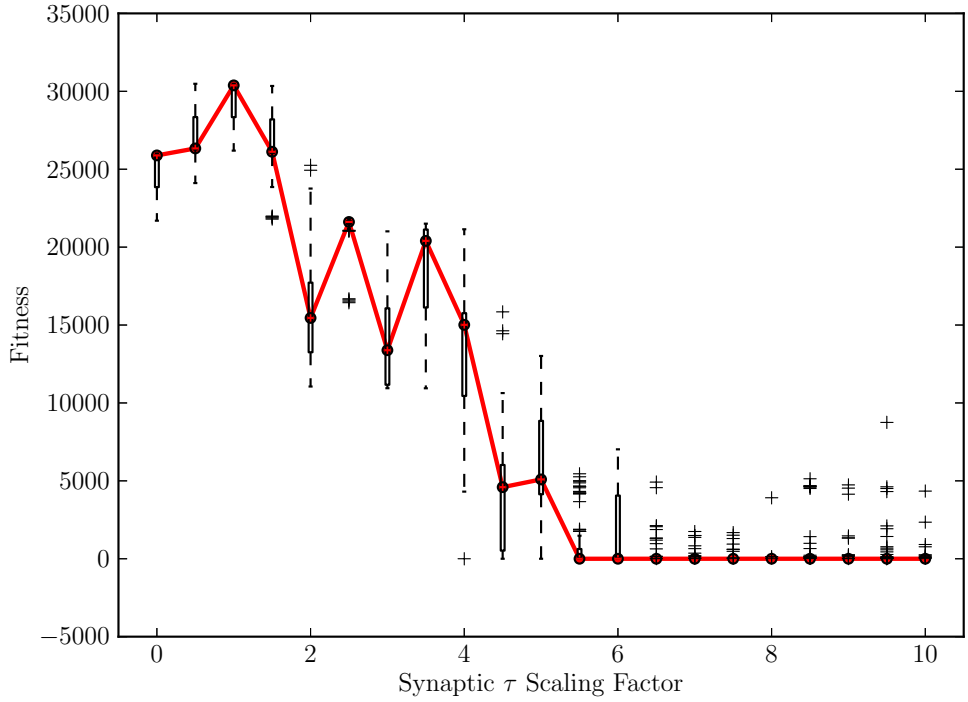
jectory. Extended delays in information propagation through the network - above those evolved for optimal behaviour that is - result in the robot being unable to effect obstacle avoidance and collide into a wall. The motor outputs then are typically observed to overreact, leading to a cycle of collisions and reflections with very poor fitness. What is more interesting is when the delays are removed and a decrease in fitness is observed. As we might intuitively anticipate, the left hand turn is made too early, causing collision with the corner as in Figure II.5.4a, but equally a collision with the far wall of the environment typically follows. The effect on the behaviour is not simply to accelerate responses, otherwise we should expect the robot to successfully avoid the far wall, but instead shows that the effect of time delays on even incredibly simple systems is anything but trivial and presents a fascinating opportunity for future analysis.

Figure II.5.5a presents box plots of the fitness spread for a range of scales, with a line plot through the median fitness values for the 100 runs of four trials. By examining the limits of the boxes, we can attain a feeling for the variability of the data. Network performance remains perfect for the evolved values, but drops off either side of the unity scale factor. Whilst the fitness function is non-smooth due to the reward and contact penalty mechanism, the fall off is fairly smooth as robots start to contact the walls more and more until they are incapable of achieving the goal areas. It is only past a scale factor of six that no trial reaches the correct goal zone.

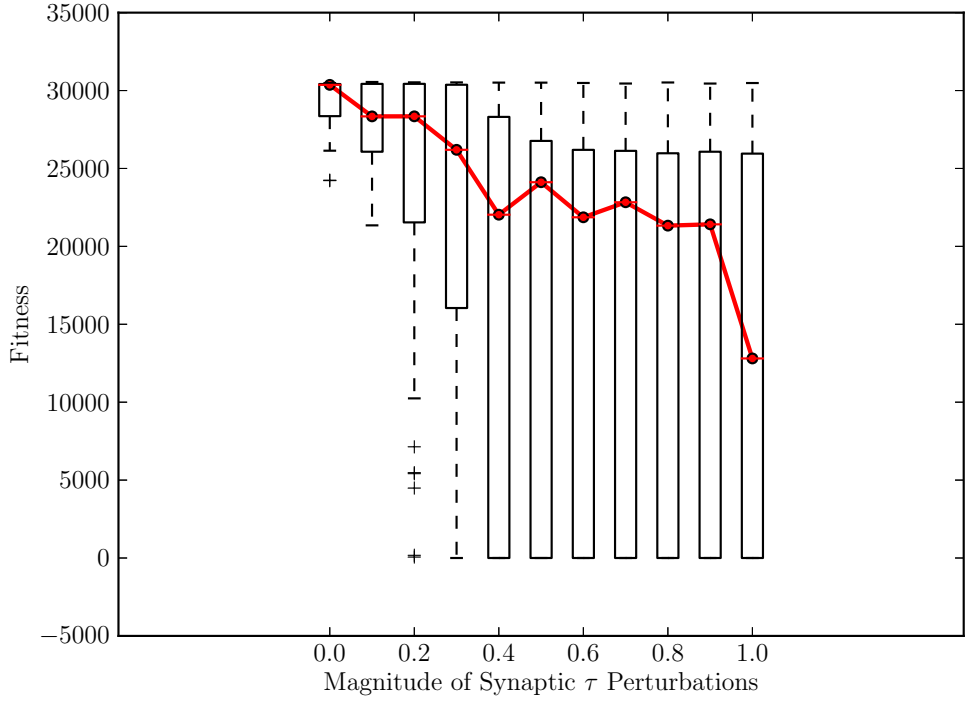
It is also instructive to perturb synaptic delays in a non-uniform manner. Figure II.5.5b shows the results of a similar trial to that previously discussed, but rather than being scaled, the value of each synaptic time delay was shifted by a uniformly distributed random number. If Γ is the perturbation factor, then the uniformly random perturbation T_p around the evolved value is in the range $-\Gamma \leq T_p \leq \Gamma$. The maximum fitness is unsurprisingly obtained when the synaptic time delays are left unaltered. As

the magnitude of the deviations from the evolved set increase, the fitness rapidly falls with a significant proportion of individuals incapable of reaching the goal area. On the whole, the effect on the behaviour is much more destructive than the scaling analysis with even very small perturbations resulting in chaotic collisions and individuals not even reaching the T-junction (qualitatively similar to Figure II.5.4c). Whilst the median of the population has a fairly linear gradient, the box plots show how for $\Gamma \geq 0.4$ the majority of individuals fail to achieve the maximum fitness. At this point the interquartile range varies across almost the whole range of possible fitness values, demonstrating the high degree of variability present. The ability for some individuals to maintain maximum fitness despite extreme values of perturbation is likely due to the random nature of the changes which may be small or otherwise affect an unimportant aspect of the network dynamics and fail to reduce the fitness of the individual.

The dependence on, and robustness to changes in, the synaptic delays in achieving a good fitness is similar to that of node time constants (τ_i) and the connection weights (w_{ij}) shown in Figure II.5.5d and II.5.5c respectively. The weights are slightly more sensitive to changes, which is hardly surprising, but this demonstrates that synaptic time delays are equally as necessary for the correct behaviour of the evolved system as the existing parameters of the CTRNN model. Further to this, we may also consider how adding delays to a CTRNN evolved for the same task would impact its behaviour. A CTRNN network was evolved within an identical experimental set up and achieved a fairly optimal fitness. Whilst there is no existing configuration of delays to modify, Figure II.5.6 shows the effect of randomly assigning delay values to synapses in the network from a normal distribution. Here the x-axis is both the mean and the standard deviation squared which defines the probability distribution from which the values are taken. Even for very small added delays of around 0.1 (only two time steps of the simulation) the fitness fall

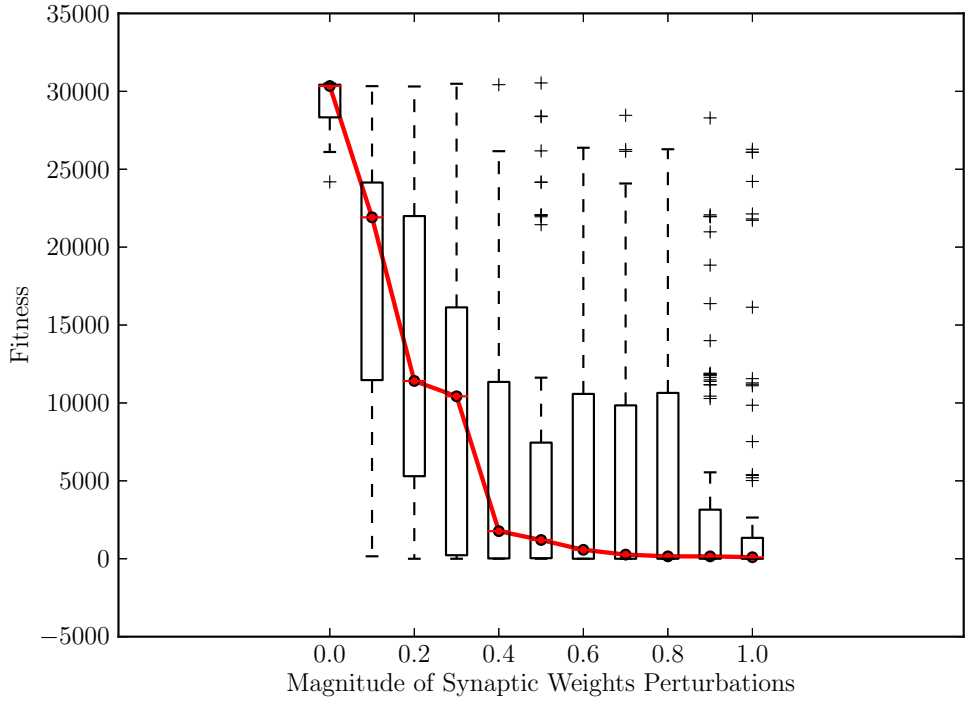


(a) Uniformly scaling synapse time delays

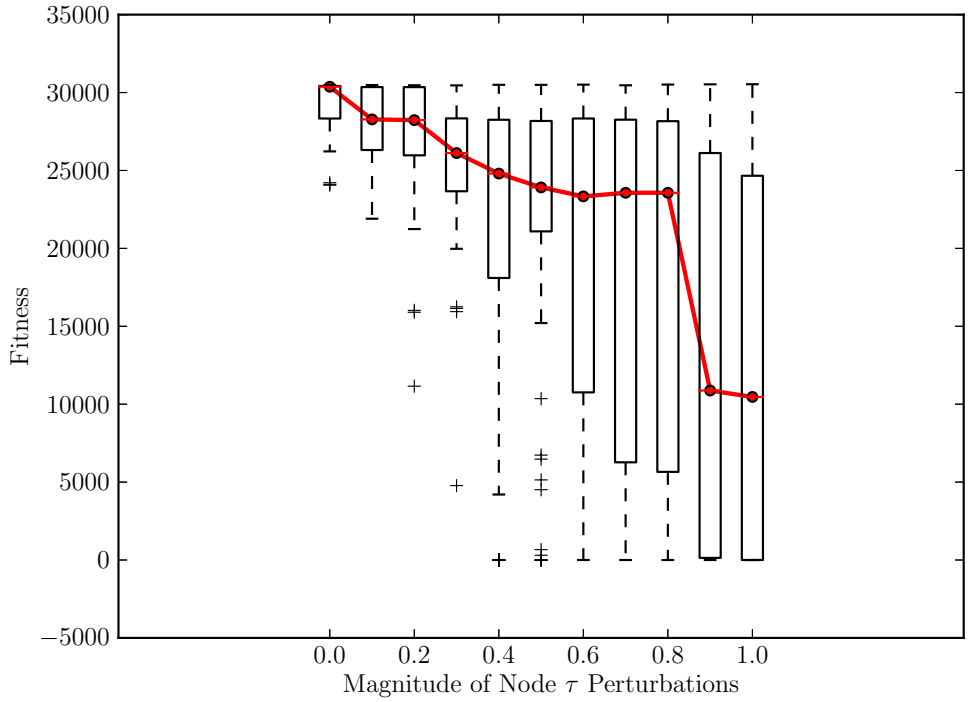


(b) Random perturbations of synapse time delays

Figure II.5.5: Sensitivity to evolved parameters



(c) Random perturbations of synapse weights



(d) Random perturbations of node time constant

Figure II.5.5: Sensitivity to evolved parameters (continued)

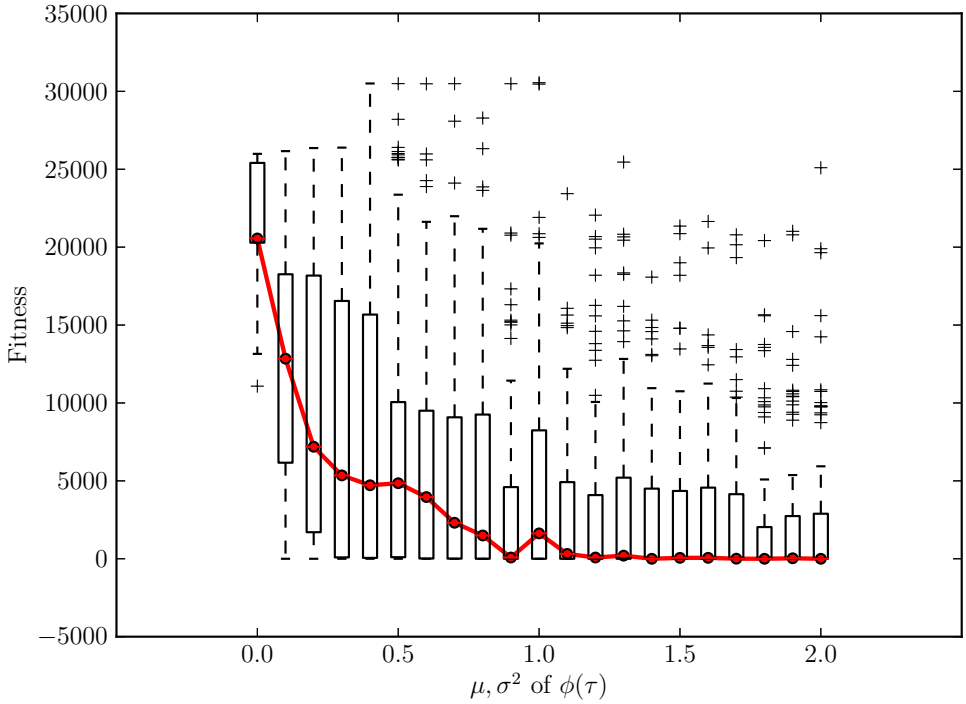


Figure II.5.6: The effect of adding synapse delays to a CTRNN

off is severe. The tolerance of the CTRNN network chosen to the inclusion of delays is very small, and has a much greater effect on the behaviour than the removal of delays had on the evolved CDRNN.

Whilst the analysis presented here is for a single specimen of both CDRNN and CTRNN networks evolved for the same task, it gives an appreciation for the sensitivity of these networks to changes in their evolved parameters. We cannot claim that these results are generally applicable, since they are particular to the task chosen and the networks analysed, but they demonstrate the reliance of a CDRNN on synapse delays for correct behaviour and the intolerance of a system evolved without delays to their inclusion. Even where a delayed network cannot be shown to present an advantage in attaining fitness, there may be situations in extreme environments where the tolerance to unintended propagation delays alone would warrant their use.

II.5.4 Conclusions

Problems in ER where learning and adaptive behaviour are sought often defy solution by discrete neural networks, but require the continuous dynamics expressed over a range of time scales by models such as the much studied CTRNN. Here, an extension to the RNN model, the CDRNN, has been applied for first time to a problem in robotic control. The addition of synaptic time delays is hypothesised to facilitate and potentially expand the capability of evolved continuous neural systems to develop adaptive and learning behaviour requiring some degree of memory. The propagation delay along neural pathways, whilst modelled in tremendous detail by computational neuroscientists, had yet to be used in ER. These additional temporal dynamics are thought to be important in biology and their inclusion may benefit the search for evolved intelligence. The impact of these delays upon the dynamic behaviour of a randomly generated network demonstrated their potential alteration of the dynamics. As it has been argued that sufficiently rich dynamics over a wide range of timescales is crucial to develop learning behaviour [45], the inclusion of these additional network dynamics presents, at the very least, an interesting prospect.

To illustrate the application of CDRNNs to robotics a controller was evolved for a simulated Khepera III robot to solve the T-junction task frequently studied within this domain. This task requires the integration of reactive obstacle avoidance with associative memory such that the robot may make a turn in the correct direction indicated by a light on approach to the T-junction. The successful evolution of this behaviour proved the capability of such networks and the resultant controller was analysed to determine the role of the synaptic delays in the governing dynamics. The evolved solutions reported are favourably comparable with the work of Jakobi [59], on which this experiment was modelled. Whilst the fitness functions and scale of the domain were different and thus direct comparison and benchmarking

is impossible, qualitatively similar behaviour was observed to known near optimal solutions. Sensitivity analysis of the evolved parameters demonstrated the reliance of the evolved systems on the presence of the delays at their specific values. A large number of trials over a wide range of perturbations to the network parameters show how the robustness with respect to changes in the evolved time delays was similar to that of the node time constants and connection weights. The importance of the time delays in the evolved solutions validates their presence and leads us to seek problems in ER unsolvable by any other means, and carry out such trials on real robots. These may shed new light on the mechanics and emergence of learning and adaptive behaviour in both natural and artificial systems.

Chapter II.6

Methods for Determining Time Delays

The preceding material developed in this Part of the Thesis has included the effect of introducing time delays into dynamic ANN and the evolution of such systems to solve a well studied example in adaptive behaviour, presented in Chapters II.2 and II.5 respectively. These experiments simply encoded the connection delays directly as additional defining parameters of the network, as discussed in Section II.6.1 below. However, one of the key research questions presented in Part I was how might we best encode connection delays beyond this simple approach. There are two sides to this question, namely the efficiency and efficacy of the representation. By adopting certain mechanisms it may be possible to minimise the increase in dimensionality of the system and so render it more feasibly solved. Note however that this is a complex issue as the characteristics of high dimensional solution spaces can be very difficult to establish, but broadly speaking a higher dimensional system is harder to search over and takes more computational effort. Equally, by intelligently designing the genotype to phenotype mapping of networks it may be possible to constrain feasible values to regions of high fitness in the solution space and achieve potentially more optimal solutions. In this

Chapter a range of methods are proposed to relate the connection delays to a spatial representation of the network which impacts both encoding efficiency and the range of possible delay values in a variety of ways. In accordance with the research questions of Part I these methods are then applied in Part III to a number of commonly studied ER tasks. This is to evaluate their success at efficiently and effectively encoding delays and how these methods might affect the range of possible solutions.

II.6.1 Directly Encoded

In wishing to explore how synaptic time delays effect and, we hope, enhance the performance of neural networks we must place their values under evolutionary control. The simplest way of doing this is to incorporate them into the evolutionary algorithm as additional parameters of the network in the same manner as the time constants and connection weights. Each synapse in the network is therefore assigned a delay value. As the size of the network increases the number of connections can tend to become very large, depending on the degree of connectivity in the network. This increases the dimensionality of the search process and may render the discovery of high fitness solutions more difficult. However, it is worth noting that this conclusion is not as easily reached as one may initially think. The structure of high dimensional fitness spaces is extremely complex, and it may well be that a higher dimensional space, despite being larger to search, is structured in such a way as to make finding high fitness solutions easier. Equally, it may be that only through the expansion of the network dynamics by adding further parameters that the network is capable of developing optimal solutions within a reasonable time frame.

During the evolutionary process, after each generation, the position of nodes in the network is subject to the same real-valued probabilistic mutation as the rest of the parameters (e.g. time constants, synapse weights, etc.).

The gaussian distribution of the mutation strength balances fine tuning of delay values whilst allowing for low probability high strength mutations to explore different regions of the fitness space.

II.6.2 Spatial Representation of Network Delays

In his Thesis, Harvey suggested that, "learning ability in some system is a behavioural level description of that system. And that the only implication for the 'working parts' of a system able to learn is that components operating on differing timescales are necessary..." [45, p.62].

In the case of CTRNNs, there are additional variables which determine the dynamics of the network, including the time-constant and bias terms [15]. In biology, the time taken for a signal to propagate between neurons is dependent on their spatial distribution and hence the geometric topology of the network as well as the nature of the connectivity between neurons. It is proposed to link the time-dynamics of nodes to the length of the connection between them, and to place the geometric definition of the network under evolutionary control. From a biological perspective there is great value in considering the geometric and structural definition of neural circuity [88].

> Just as information in a map is held by such spatial properties as physical distance, the physical structure of cortex encodes information. With geometric principles of information processing the information is held in the three-dimensional pattern of neural connectivity. As constructive factors play a central role in building this physical structure, they also shape the representational properties of cortex. Building neural circuits with directed growth thereby builds the brain's representational properties.
>
> These spatial properties of representation are largely lost in the traditional connectionist network because of the way the con-

> nectionist neuron integrates information, typically summing its input and sending a (perhaps graded) output if some threshold is exceeded. This makes the entire cell the basic computational unit. In contrast, biological neurons are thought to segregate into subregions that function as autonomous processors.

In determining delays by considering the spatial configuration of nodes in the network, we limit their possible values. In this way, the solution space for the network is reduced. This could aid the evolutionary search as a reduced search space is typically easier to explore, but could equally prevent the location of high fitness combinations outside the geometric limitations of the network. Consider a simple fully connected three node network in which the connections between the nodes must form a triangle. If the edges of the triangle are a,b and c and we consider all possible shapes the Triangle Inequality Theorem states that:

$$a + b \geq c \tag{II.6.1}$$

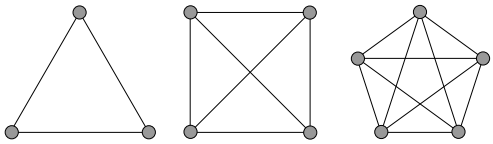
Where $a \leq b \leq c$.

If the edge lengths were normalised such that $(a, b) \in [0, 1]$, and $c = 1$, then $a + b = 1$ is the isoline of triangle validity. Thus, half of the possible length values for a and b from 0 to 1 are no longer valid, representing a corresponding 50% reduction in the delay variable space.

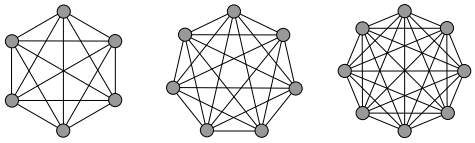
Taking this analysis further, it can be demonstrated that as the size of the network increases, the proportion of the space in which valid network length configurations lie becomes very small. Figure II.6.1 presents a series of simple polygonal networks with n nodes.

For a fully connected (but not self-recurrent) system of n nodes there are c connections. Where:

$$c = n + \frac{n}{2}(n - 3) \tag{II.6.2}$$



n	3	4	5
c	3	6	10



n	6	7	8
c	15	21	28

Figure II.6.1: Primitive Network Geometries

As previously discussed, for a 3 node system this is simply determined. A two node system can have any range of connection lengths in the range of zero to infinity, but for more complex shapes the distribution of connection lengths which lead to valid geometry becomes difficult to determine. There are then c equations which be satisfied for the shape to be valid of the form:

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - l_{ij} = 0 \tag{II.6.3}$$

For each of the c connections where the start node i, end node j with positions (x, y) and length l_{ij}. We would like to explore all possible unique configurations and not similar shapes. For this reason we shall consider possible lengths for $l_{ij} \in [0, 1]$, but require that at least one connection has a unity length.

These networks represent a non-linear system of equations which is complex or impossible to solve directly as they are overdetermined with more equations than variables, leading us to a numerical approach. Our object is to determine some of the properties of the c dimensional space in which a point represents a unique set of connection lengths, specifically as to whether

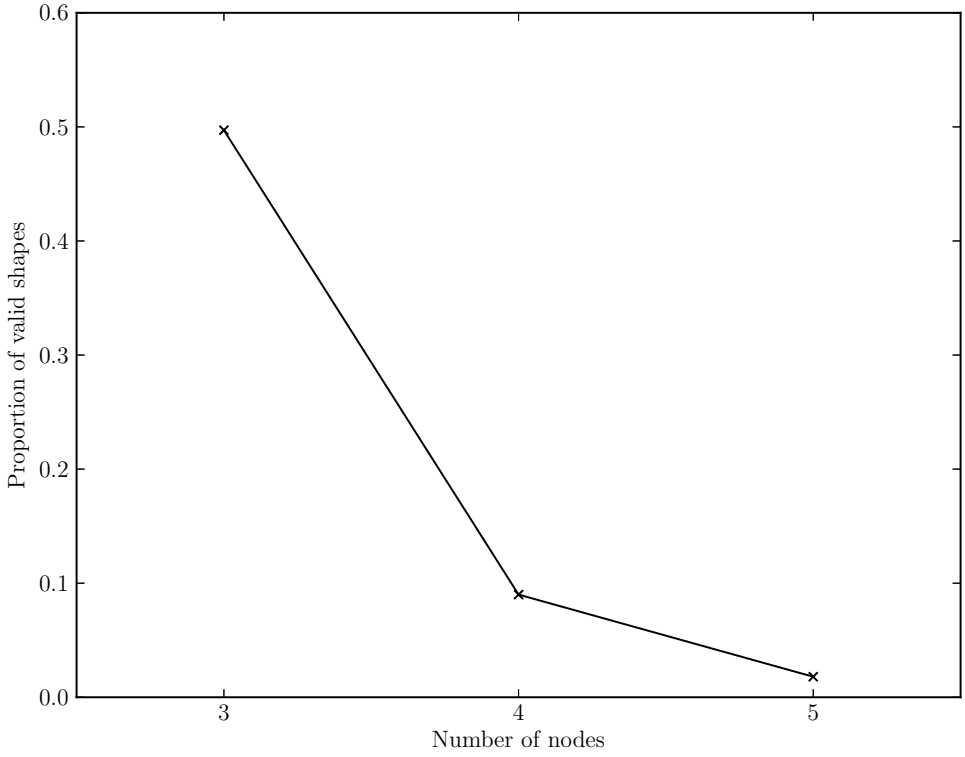


Figure II.6.2: Proportion of Valid Configurations

this can be assembled into a network where the spatial constraints are maintained. A study was conceived whereby the system of equations was evaluated using a non-linear numerical method,[1] for each point of a uniformly spaced c dimensional grid. This method required an initial guess from which the algorithm proceeds. In this case, the starting point was taken to be a regular polygon formed with sides of unity length. The results of each point were then evaluated to determine if that configuration of lengths could be assembled into a valid spatial arrangement satisfying the error. The number of successful solutions was divided by the total number to obtain an estimate for the proportion of the volume in which valid arrangements existed. Even for a small number of nodes, as the number of connections increases significantly, even a very coarse grid becomes computationally infeasible both in terms of duration and memory usage. The acceptable error in connection lengths for solutions assembled by the numerical solver was taken to be equal to half the interval of the grid. Results of sensitivity analysis indicate that this typically over estimates the number of valid solutions and that as the density of the grid increases, the result descends to the true value.

The results of this investigation are shown in Graph II.6.2, for arrangements of 3, 4 and 5 nodes with a sample grid of size 200, 15 and 5 receptively. This involved the solution of 12697539 (119401, 3861089 and 8717049 respectively) instances of the equations. Such is the nature of the increase in solutions required (as the number of possible configurations, without removing similar configurations, is l^c where l is the size of the grid) that going beyond this level is very difficult and tedious for little gain. It is also worth considering that these results are for fully connected regular networks which are considerably denser and more constrained than the majority of networks under evolutionary design. It is impossible, however, to characterise the performance of real networks under this analysis, given the open ended and

[1]The Python module `scipy.optimize.fsolve` was used, see `www.scipy.org`.

stochastic nature of the evolutionary process which is capable of generating highly complex and irregular network configurations. Whilst this is true, we can certainly apply the general trend of the data shown in Graph II.6.2 to evolved networks. As the size and connectedness of the network increases, the possible configuration of connection lengths and correspondingly time delays are increasingly constrained until only a very small portion of the possible search space be explored.

This presents a very interesting line of enquiry. The proposed methods for determining synapse time delays from spatial properties of the network placed under evolutionary control is (albeit loosely) biologically inspired. For realistic sized networks, the distribution of time delays has been shown to be significantly constrained. For the case of the pentagonal network shown in Figure II.6.1 only around 2% of the possible range of synapse time delays may be represented. We have seen in the sensitivity analysis of Chapter II.5 how the values of delays are crucial to the behaviour of evolved systems. We might therefore assume that this spatial representation of delays encoded into the position of the network nodes might severely restrict the evolution of desirable behaviour. Equally, or perhaps more, interesting is the case where they do not. Should there be a neutral or positive effect by expressing delays in this manner this must influence our conclusions as to how the delays effect the behaviour, how easy good configurations are to find, whether the biologically inspired methods are in some way restricting the search to high fitness regions, and whether this form of encoding may be shown to be more efficient. These questions will be discussed in Chapter III.1 in the context of the results of the examples presented.

II.6.2.1 Assigned Network Geometry

Both of the methods described below represent ways of placing the position of nodes in a network under evolutionary control. This however requires that

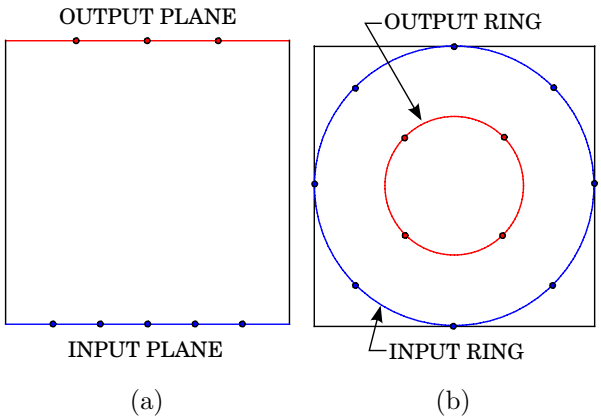


Figure II.6.3: Initial Network Geometries

the network is assigned spatial properties prior to the subsequent modification. Typically when networks are drawn they are presented for maximum clarity in their illustration of network connectivity and structure. The analysis of Section II.6.2 shows us how crucial the distribution of nodes is to the range of possible delay values which can be represented in this configuration. Care should therefore be taken to ensure that the initial layout with which a network is assigned is appropriate to the task and does not over constrain possible solutions. It is worth noting that no dimensions are considered here, it being simply the arrangement of the nodes and the relative modification of their position by the methods described which determine the delays in the network. For convenience, we shall describe nodal positions in a Cartesian coordinate system (x,y,z) where the initial arrangement of nodes lies between zero and unity on an arbitrary scale.[2]

Figure II.6.3 shows a range of possible network layouts. Often networks are shown as having a Feed-Forward structure, with the possibility of recurrent synapses naturally, with different stages of the network arranged in the

[2]N.b. The nodes are not constrained to remain within the bounds of $0 \leq (x, y,) \leq 1$ as the methods described later may move them outside of these to lengthen connections and increase delays.

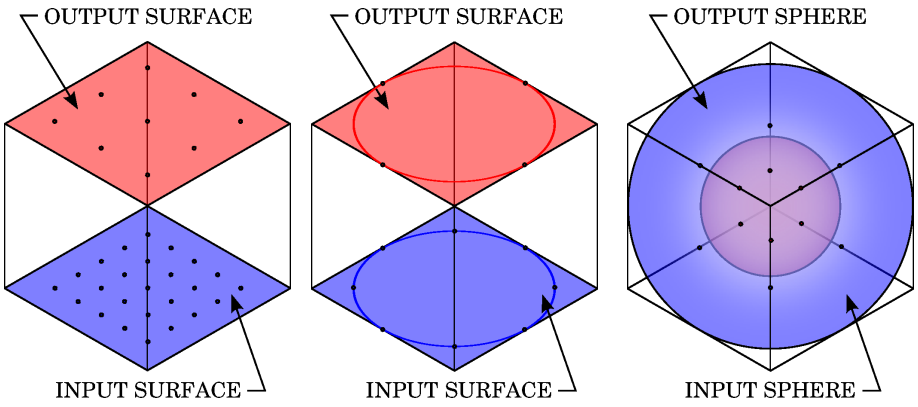


Figure II.6.4: Possible 3d Arrangements

general direction of information flow as shown in Figure II.6.3a. Here, nodes which act as inputs to the network are arranged equidistantly along an input plane (y=0) with the outputs of the network arranged at the other end of the space (y=1), forming an arrangement of nodes n where $x_n, y_n \in [0, 1]$. Any hidden nodes would then be located on a intermediate plane between the two (y=0.5). With a non-recurrent minimal genotype and synapses only between input and output nodes, the connections are of broadly equal length. This approach is generally suited to the majority of scenarios, especially those where there is a definite structure to the network i.e. in an obstacle avoidance scenarios where proximity sensor input is processed by the network to generate motor outputs.

Alternatively, one may consider a scenario where input and output nodes are arranged on concentric rings of differing radii (r=1.0 and r=0.5) focussed on a point in the centre of network space (x,y = 0.5,0.5), shown in Figure II.6.3b. In contrast to the first proposal, this arrangement of nodes appears more suited to networks without such a clear structural definition. Consider a locomotion task, where all nodes in the network directly control effectors but also can receive sensor inputs. Assigning geometry in such a formal

layered structure as above is tenuous and a more mixed arrangement, but with underlying order, may be more suited.

Both of the structures proposed are two dimensional simplifications of the three dimensional shapes found in biological circuitry. This assumption is made for simplicity; however, if it can be shown that by limiting ourselves to 2d structures, we over constrain the possible configurations of delays, it may be worth considering three-dimensional structures of the type shown in Figure II.6.4.

II.6.2.2 Position Encoded

Previously we have discussed the possible justification and advantages of determining the dynamics of a network through its spatial properties, and how this may constrain the focus of any optimisation algorithm. In this Section, the simplest method for allowing an evolutionary algorithm to modify the geometry of a network is presented.

In this methodology, the minimal genotype must be assigned initial geometry as previously discussed. As the NEAT complexification process proceeds, additional hidden nodes may be added to the network structure. This happens through the splitting of an existing synaptic connection and, in the standard NEAT method, has the effect of adding a non-linearity (in the form of the node's sigmoidal activation function) into the dynamics of the network. In this proposed modification to the standard scheme, the new node is placed at the average of the split synapses' input and output nodal original positions. These nodal original positions are persistent through multiple generations and both the modification by the CPPN, and the updating of synaptic time delays, must take place prior to each simulated run.

Each node in the network is now assigned two additional parameters, namely (x,y)-coordinates in two dimensional Cartesian space as described in Section II.6.2.1, which are mutated during the course of the evolution-

ary process in the same manner as in the directly encoded method. The distribution of possible mutations is pre-determined to provide appropriate displacements given the assigned geometry of the network.

The absolute distance between nodes varies between all of the nodes in the network, most of all when a new node is added which halves the particular synaptic lengths. As the spatial distribution of the nodes is arbitrary, the synaptic time constant must therefore be set in relation to an initial starting delay common to all connections which becomes a fixed parameter of the simulation. This original value is then modified by the relative change in the length of each synapse depending on the modification of nodal position, thus:

$$\tau_{ij} = \Gamma \sqrt{\frac{(x_{i_n} - x_{j_n})^2 + (y_{i_n} - y_{j_n})^2}{(x_{i_o} - x_{j_o})^2 + (y_{i_o} - y_{j_o})^2}} \tag{II.6.4}$$

Where τ_{ij}, is the synaptic time delay between nodes i and j, Γ is the fundamental synaptic time delay common to all connections, the coordinated of node i and j are (x, y), n indicates new node positions and o indicates the original starting positions of the nodes.

In this way, the spatial representation of the nodes in the network and therefore the length of synapse time delays is placed under evolutionary control. The configuration of nodal positions restricts the proportion of possible values that the delays may take, based upon the complexity and structure of the network. Whether or not this affects the ability of the evolutionary algorithm to attain high fitness solutions is discussed in Chapter III.1.

Beyond the linear determination of delay values it would be simple to consider a non-linear mapping to allow small changes in connection length to make significant influences of the value of delay.

II.6.2.3 Pattern Encoded

In his comprehensive work on the evolution of biological brains, Striedter describes the structure of typical mammalian brains, and how most brain

regions exhibit structural regularities [109]. Typically seen are laminar regions, where densely connected laminations of neurons run parallel to each other, with sparse interconnections allowing the distribution of 'processed' information outside of the densely connected channels (often linking stimuli to effectors) [109]. Laminae are thought to be easy to evolve and important structurally in the neural processing of information, and thus it would be of interest to develop a system capable of generating comparable structures under evolutionary control and if this would result in better evolved solutions. Striedter illustrated a simple model of laminae development through neuron growth over two perpendicular independent linear gradients [109].

Stanley et al. have developed an interesting method for the artificial evolution of large scale artificial neural networks using CPPN [106], introduced in Chapter II.1. CPPNs are different from ANNs, but may be evolved in the same way through their HyperNEAT method. NEAT has also been used to evolve both two and three-dimensional patterns.[3]

The proposed concept is to bring these ideas together and place a two- (or three-) dimensional pattern under evolutionary control and use it to modify the geometric representation of a recurrent neural network. Whilst we hypothesise that delayed dynamic networks have the potential to develop interesting solutions with their extended dynamics over traditional dynamic neuron models, the additional parameters increase the search space, potentially rendering good solutions harder to find.

In order to place this geometric representation under evolutionary control, the nodal positions of the evolved CDRNN are to be modified by a co-evolved CPPN. This means that each individual in the population effectively consists of two genomes; one for the neural network and the other for the pattern network. The pattern network is then evaluated to modify the neural network as discussed later. This is shown graphically in Figure II.6.5.

[3]See http://picbreeder.org/ and http://endlessforms.com/.

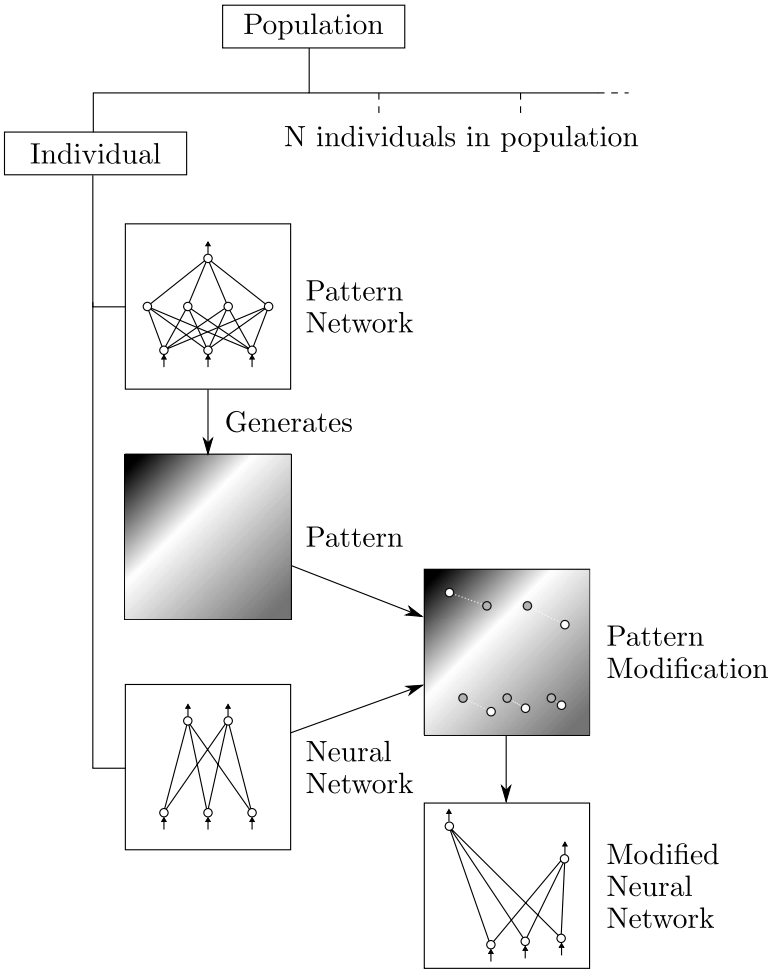


Figure II.6.5: Dual Genome Architecture

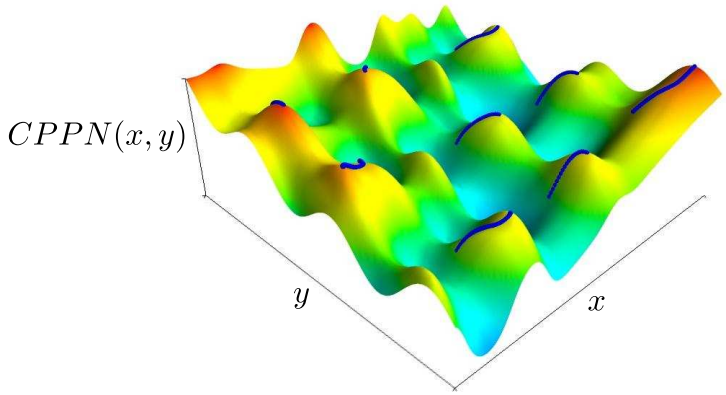


Figure II.6.6: Hill Climbing Nodes over CPPN Surface

Whilst this too increases the complexity, and thus the difficulty, of the evolutionary process, the CPPN search space is independent of complexity of the CDRNN and for larger problems will present a more efficient encoding of the additional evolutionary parameters.

A CPPN evolved in parallel with a particular CDRNN generates a pattern when sampled over the input space of (x, y)-coordinates (for the 2D case considered initially). CPPNs are capable of generating both simple and complex patterns which are continuous in the range of possible inputs, where $(x, y) \in \Re^2$. The two-dimensional intensity pattern may be considered as a three-dimensional surface, over which it is proposed that the network nodes hill climb for a number of iterations as shown in Figure II.6.6. In this way the CPPN may alter the position and thus the time constants of the synaptic connections, so the CPPN genotype indirectly encodes the additional evolutionary parameters of the CDRNN with respect to the more traditional CTRNN form. Patterns may cause clustering or dispersement of neighbouring neurons which typically process related information, and thus create fast or slow connections between them. It may therefore be possible to create structures analogous to ocular dominance columns seen in mammalian visual cortices, and other common neurological features.

The aim of this whole procedure is to place the synaptic time constants under indirect evolutionary control which may significantly alter the time delays, in order provide network dynamics with a wide range of time scales to encourage and enable learning and adaption. This requires that the default synaptic time delay (Γ) is set to an intermediate value at which fit solutions are capable of being generated, but that the evolutionary process can alter the value either way to effect significant changes in the dynamics of the system. As the changes to the time delays are proportional to the change in synaptic length, a larger Γ will provide more noticeable alteration in the dynamics of the network, but at the expense of embedding a large delay in the system which may render fit solutions difficult to generate. One side effect of this methodology may be the requirement for smaller time steps than would be required for the stable integration of traditional CTRNNs, thus imposing greater computational demands during the evolutionary process.

The Python implementation of the NEAT Methodology introduced in Chapter II.5 was modified for the purposes of demonstrating the ideas proposed. A key concept in the proposed model is to simultaneously evolve both a neural network and a CPPN using the NEAT method. To this end, each organism created maintained two network instances which were identified as a neural network or pattern network. As they existed within the same organism, fitness attributed to the organism was effectively applied to the combination of the pattern and the neural network. More complex forms of fitness sharing may be an interesting avenue for future research. Each CPPN node maintained a type which determined the form of the activation function applied, as a wide variety of activation functions is central to the CPPN-NEAT method. As each node is created, it is randomly assigned an activation function from the finite set of possibilities. Possible functions were: linear, sigmoid, sin, cos, tanh and gaussian. The CPPN minimal genotype had x,y, and $bias$ input nodes, and three hidden nodes

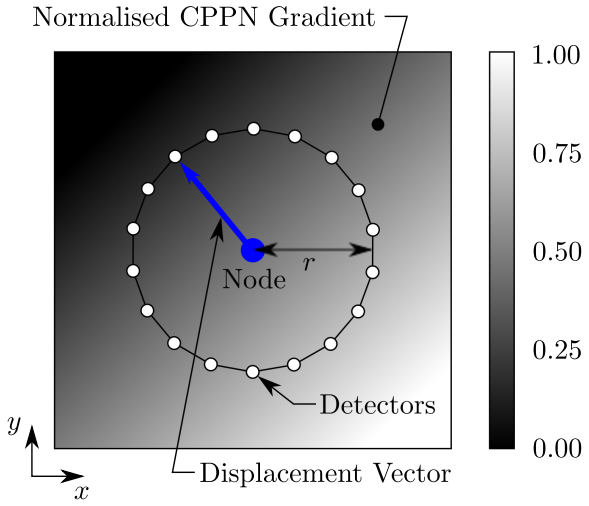


Figure II.6.7: Finding the Gradient Direction Vector

each initialised with a $\tanh(x)$, $\sin(10x)$ and gaussian activation function respectively. This was in an attempt to generate a wide variety of patterns across the population throughout the initial generations, without having to wait for the evolutionary process to develop mature and complex networks. Initial patterns may range from simple one dimensional gradients to highly complex repeating functions.

At the start of each generation, nodes of the delayed network are assigned initial positions as described previously in Section II.6.2.2. For a set number of iterations the nodes hill climb, selecting the direction which gave the greatest improvement in the value of the CPPN. Around each node, the CPPN was sampled at a set number of detector points on a circle of a set diameter as shown in Figure II.6.7. In the results reported here, there were 18 detectors at 20° intervals, on a radius of 0.01. If the maximum value of the detectors is greater than the current position of the node, then the node is moved toward that detector by a given amount.

In order to allow a wider range of modification to the nodes, the position that each node moves is not constant, but rather is related to the gradient of the CPPN. At the start of the process the CPPN is sampled over the

distribution of nodes in a grid to approximately determine the maximum and minimum values of the CPPN in this region. The distance moved by each node in the direction of the maximal detector unit vector is equal to the difference in the value of the CPPN between the detector position and its current value, divided by the overall range of CPPN values. Thus a node on a greater gradient will move further, allowing a greater range of movement of the nodes in the network over multiple iterations. The gradient is normalised as the range of values for different CPPNs may vary over several orders of magnitude and it is the pattern which is of interest, and not the absolute values. In this way, the node climbs towards any local maxima, altering the layout of the neural network.

The configuration of initial nodal positioning is key to the development of good solutions. The work presented herein has only thus far considered a simple planar distribution of nodes, but would do well to consider other arrangements. It is natural, and biologically justified, to consider the application of this method to three-dimensional instead of planar geometries, and this is a very simple extension of the algorithms presented. The evolved CPPN should have an additional input node, such that $f = \text{CPPN}(x, y, z)$ with a Cartesian coordinate system (for cylindrical or spherical domains a polar-coordinate system may be more suitable). The simulated detectors would then be distributed on a spherical surface, with the remainder of the algorithm little changed. This would be of great interest as the number of possible configurations is greatly increased and the evolutionary process may take advantage of the additional complexity of network structures made possible by the extension into 3D.

II.6.3 Efficiency of Different Encoding Methods

We have discussed how each of the different methods for encoding synaptic time delays into an evolutionary algorithm may constrain the range of values

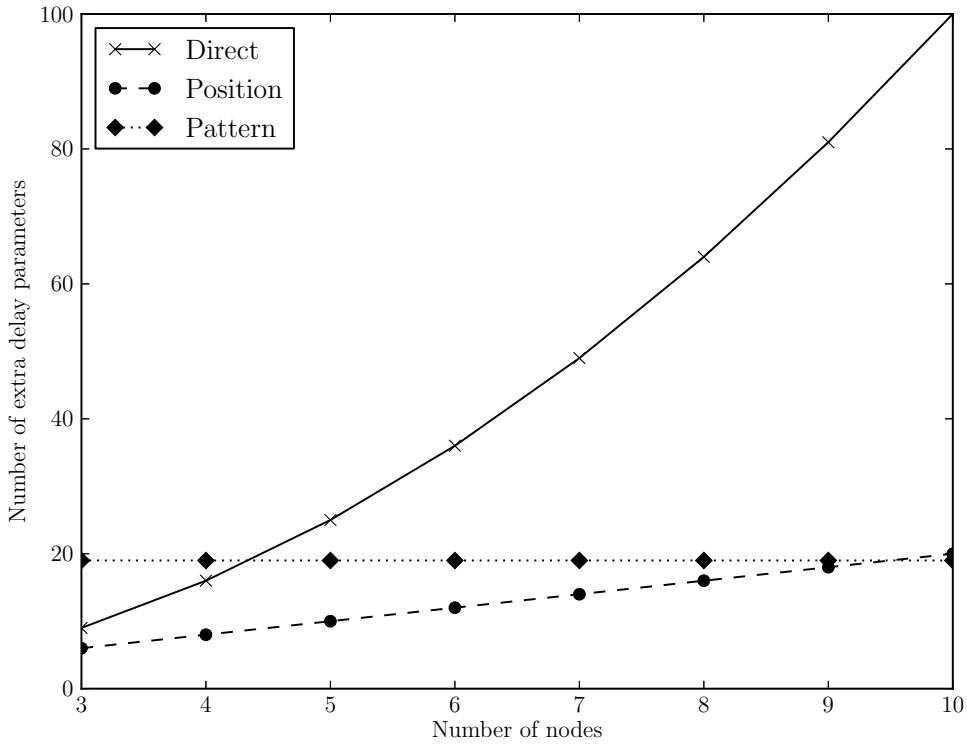


Figure II.6.8: Encoding Efficiency for Different Methods

these delays can take. Also, we have considered how, by increasing the dimensionality of the search space by adding further parameters which define the network, we may be rendering the process of evolutionary optimisation more difficult. Care is traditionally taken in ER to minimise the structure of networks, and to exploit any symmetry or repetition in morphology or control architectures, to reduce the number of parameters under evolutionary control to a base set required for the emergence of the desired behaviour.

Three different methods of determining delays have been proposed, and we must therefore predict the increase in search space dimensionality for each. The relationship between this increase and the complexity of finding fit solutions is extremely complex and impossible to fully define as it is highly subjective as to the form of the networks, the nature of the task and the design of the fitness function. It would also require the comprehensive evaluation of extremely high dimensional spaces which is not only computationally unachievable, but impossible given the nature of the real valued open ended evolutionary algorithm used throughout this research. Nevertheless, it is an indicator of complexity and therefore may be used to make tentative judgements regarding the efficiency of each method in encoding the additional delay parameters.

Given the open ended nature of the NEAT algorithm used, and the resulting variable length genotypes of evolved solutions we must make some assumptions in order to quantify this relationship. This is necessary as each method may variably rely on the number of neurons or synapses in a network which is highly variable given the initial minimal genotype and probabilistic complexification of the evolutionary process. Here, we shall consider fully connected and self recurrent regular polygonal networks of the form shown in Figure II.6.1, but with the additional synapses. If the number of neurons in the network is n, then the number of synapses c is simply n^2.

When the time delays are directly encoded (see Section II.6.1) there

is one addition parameter for each synapse. By contrast in the position method (see Section II.6.2.2) the n^2 delay variables are uniquely determined by controlling $2n$ parameters i.e. (x,y)-coordinates for each of the n neurons. Whilst modifying positions of the nodes using a pattern uses only the parameters required to define the CPPN to determine the delays, thus rendering it independent of the size of the network. For the pattern network there is one weight parameter per connection and one variable activation function per node, making a total of $(n + c)$ parameters. For the research presented throughout this Thesis, a minimal genotype was devised for the CPPN which incorporated a range of different activation functions and hidden nodes to allow the generation of interesting patterns even during early generations. This genotype had 7 nodes and 12 synapses, which is therefore fully defined by 19 evolutionary parameters. This may increase of course as the CPPN is subject to the same open ended evolutionary process as the other network and may therefore increase in complexity throughout the process, requiring additional parameters. Figure II.6.8 shows these relationships evaluated for a range of network sizes commonly seen in ER investigations. As can be seen, the direct method follows the n^2 relationship and quickly requires the most evolutionary parameters of any method. The position encoding has a defined linear relationship, with the pattern encoding remaining constant, independent of the size of the network. For small networks the position encoding requires the fewest additional parameters, but when $n = 10$, it over takes the pattern method. As n increase, the number of parameters for the direct encoding method rapidly explodes. Given the underlying assumption that a large number of additional parameters is typically harder to optimize than a smaller number it can be seen that the two spatial representations proposed can provide a significant advantage over the direct method.

Other factors governing the computational performance of different methods are the amount of memory and the computational effort required for

each. Again, specific meaningful values depend greatly on the probabilistic nature of the evolutionary algorithm and are hard to obtain. However, both the direct and position encoding methods require roughly the same amount of memory. The position encoding requires a greater degree of post-processing than the direct in the nature of calculating the deviation of connection lengths from the initial assigned positions and therefore requires slightly more computational effort. The pattern encoding technique requires significantly more memory and computational effort not only for the storing, evaluation and evolution of the co-evolved CPPN, but also in the modification algorithm described in Section II.6.2.3. Evaluating the CPPN in a grid to normalise the pattern and hill climbing the nodes of the network across it requires a significant effort. This can cause an evolutionary run using this technique to perform noticeably slower than the others presented here.

As to whether the spatial representations are in fact more efficacious than the direct method of encoding delays, this can only be determined through experimentation. By evaluating a range of tasks with each of the methods presented we can then judge as to the relative merits of each as a trade-off between the constraints imposed on the range of possible values and the performance each method can achieve.

II.6.4 Comparing the Efficacy of Different Encoding Methods

To evaluate, analyse and make meaningful conclusions regarding the inclusion of time delay dynamics in continuous recurrent neural networks and the relative merits, or otherwise, of the different methods of determining these delays within an evolutionary algorithm, we must be able to evaluate each possible configuration under a range of scenarios. In common with the approach taken to the simulation modelling of Section II.4.3 we must

ensure similarity and compatibility of simulations that we may be justified in comparing their results.

The Python implementation of NEAT used, and heavily modified, throughout this work was configured to allow determination of the neuron model and the delay method through the configuration file loaded at runtime by the algorithm. The appropriate mutations were then applied during the reproductive process depending on which configuration was used. In all other respects the code responsible for the behaviour of the system under investigation were identical. Differences in performance must therefore be due to the effects of the different delay methods and the non-deterministic stochastic nature of the evolutionary process. Where feasible, multiple runs may be undertaken to estimate of the variability of results due to the initial sampling of the search space and successive probabilistic mutations and reproductions that are at the core of the evolutionary algorithm.

II.6.5 Conclusions

To be able to investigate the effect of adding time delays to synaptic connections within neural networks, we must develop methods by which they can be controlled by the evolutionary algorithm. The simplest and most obvious method is to encode the connection delays directly as further parameters describing the dynamics of the network. However, this fails to take advantage of potentially interesting solutions suggested by a spatial representation of the network. By applying this biomimetic approach we can constrain the configuration of possible delay values and reduce the dimensionality of the search space. This may render the optimisation process less complex than if the delays were encoded directly, but has the potential to exclude high fitness delay combinations due to the geometric constraints enforced.

Two methods for determining the spatial representation of the network have been proposed which essentially deform the network with respect to

an initially assigned geometry, and thus modify delay values in a manner proportional to the changes in synaptic length. This is achieved by either mutating the position of nodes in the network directly, or by modifying the geometry with a co-evolved pattern network in a bioinspired manner. Both of these techniques have potential advantages and disadvantages with respect to their computational efficiency and the reduction in dimensionality of the search space compared to the direct method.

The evolutionary algorithm used throughout this work has been configured to allow the side-by-side comparison of all possible configurations of neuron model and delay evaluation technique. This will allow, after the examination and analysis of repeated trials of multiple experiments, to reach a quantifiable conclusion as to the potential advantages or disadvantages of including delays in these systems for each of a range of possible fields of applications and the efficacy of each method for calculating the delays.

Part III

Results and Analysis

Chapter III.1

Introduction

Previously, non-linear dynamic networks with time delays have been introduced and proposed as architectures for evolved adaptive behaviour. Our hypothesis is that the richer dynamics introduced by the delays into the behaviour of the system may enhance their performance. In order to accept, or reject, this hypothesis this Part presents results for a variety of tests largely inspired by nature and the scientific study of animal behaviour.

In Part II of this Thesis the concept of including simply modelled transport delays in the connections of CTRNNs was analysed and then implemented. Chapter II.2 exposed the increased dynamic capabilities of networks modified in this manner and elucidated the special circumstances under which even a single node can oscillate with no external input. Through Chapters II.3 and II.4 the tools prerequisite to an exploration of the research questions, defined in Part I, have been developed and include a novel approach to computationally minimal simulation of a quadruped robot and physically derived 2D wheeled robot models. Driven by the research question to find ways of harnessing time-delayed networks in EAs, Chapter II.6 explores a variety of ways to encode network delays in an evolvable genotype; taking inspiration from biological systems and the spatial properties of neural networks. This Part applies each of the encoding methods developed to a

range of commonly studied tasks and compares the results to the behaviour of an unmodified CTRNN in accordance with the research question.

Continuous systems modelled on higher order neural networks have demonstrated a wide range of adaptive behaviour. Specifically, results for simulated chemotaxis and T-test (see Chapter II.5) have been well documented but demonstrate such core aspects of adaptive behaviour that their inclusion is still warranted.

The results presented herein are validated insofar as it is possible through the statistical treatment of data from repeated runs of each configuration. For simulations of an extended or computationally intensive nature a high number of repeated runs may be infeasible given the time constraints of this research. The level of confidence for each set of data is presented and discussed in the context of the validity of conclusions as to the benefits (or otherwise) of including delays in these systems.

From the data it has been possible to draw a number of conclusions regarding the inclusion of time delays in dynamic neural systems and the variety of methods which have been developed to determine them. The results show that delayed dynamic neural systems are at least as capable as traditional CTRNNs in the experiments below. The increased dynamic response of such systems enables the evolution of very small pattern generating circuits at a level not previously possible. This allowed a three neuron circuit to near optimally control the gait of a single gait. Further dynamics inherent in these systems made qualitative improvements to behaviour in other experiments which were not captured within the fitness functions used. However, in the adaptive behaviour tasks investigated no such improvement was seen. This could be simply that the tasks did not require such complexity for optimal solution, and it is in the development of such a task that presents the best immediate avenue of future research.

III.1.1 Hypotheses Under Test

Specifically, there are two hypotheses that we wish to test through statistical analysis of experimental results. First, that by including delays it is possible for dynamic networks to express behaviour previously unattainable for the solution of problems in adaptive behaviour and legged locomotion. Aside from examples in the development of gaits where the increased dynamics of the system may directly map to an increased ability to perform in a task, it is difficult to demonstrate increased fitness for a network of delayed type versus an already optimal known solution. Where possible the fitness function may be designed to highlight such differences, but typically delayed dynamics may often result in a retarded response to stimuli and thus effect a small reduction in fitness (depending on the fitness function). CTRNNs have proven to be highly capable control structures, and finding an example of adaptive behaviour which defies solution by non-delayed systems but is solvable with delays is difficult and requires a great deal of trial and error.

Secondly, that including delays increases the degree of information processing in the network connections. By enabling information storage that would have otherwise required the introduction of further network nodes, smaller networks than previously possible may be capable of achieving the same behaviour. This is much easier to test as it is still possible to discriminate in terms of the network structure between results with statistically similar fitness profiles. Where difficulty lies is in the correct design of minimal genotype, so that the initial structure is not already capable of performing optimally.

Beyond this, we wish to evaluate the performance of each of the methods for determining time delays developed in Chapter II.6. We anticipate that for simple tasks a position encoding may present some advantages, but that for more complex structures this representation may limit the possible configuration of delays, preventing optimal results from being reached.

III.1.2 Selection of an Appropriate Statistical Test

Since there are four independent groups, a test for k independent samples is called for. The fitness values for each group are continuous and on an ordered scale and so the assumptions for the Krustal-Wallis test are satisfied. As the number of groups is greater than 3, the sampling distribution is well approximated by χ^2 with $df = k - 1$ for a given significance level; $\alpha = 0.05$ was chosen in this case. The hypothesis is rejected if the calculated value of KW is so large that the probability associated with its occurrence when H_0 is true is equal to or less than $\alpha = 0.05$, corresponding to $KW \geq 7.82$.

We can test the significance of individual pairs of differences by using the following inequality:

$$|\bar{R}_u - \bar{R}_v| \geq z_{a/k(k-1)} \sqrt{\frac{N(N+1)}{12} \left(\frac{1}{n_u} + \frac{1}{n_v} \right)} \qquad \text{(III.1.1)}$$

If this is true, then we can reject the hypothesis that the distributions to which each sample belongs have the same median value, $H_0 : \theta_u = \theta_v$, and conclude that $\theta_u \neq \theta_v$. The value of $z_{a/k(k-1)}$ is the abscissa value from the unit normal distribution, above which lies $a/k(k-1)$ percent of the distribution.

III.1.3 Limitations

The statistical test selected above was chosen for its ability to efficiently discriminate even with low numbers of samples. Given the overriding objective defined by the research question, to evaluate the effect of introducing time-delays into CTRNNs, there is an inevitable compromise in the division of computational effort. By exploring the application of delayed systems to a variety of commonly studied tasks in ER the number of experimental repetitions, even for specifically designed computationally minimal simulations, must be limited. Because of this, and due to the finite time allowable

for experimentation following the developmental work described in Part II, the number of runs achieved in this Part is low. Whilst application of the Krustal-Wallis test elucidates sufficient statistical confidence to support the conclusions drawn from the work in this Thesis, the limitations of restricted repetitions must be appreciated. To build upon the foundations developed in this Thesis in the future further repetition of the experiments in this Part may be justified to provide a stronger evidential basis.

Chapter III.2

Adaptive Behaviour

Much of the work in ER has been focussed on the synthesis and analysis of adaptive behaviour from fundamental building blocks, be they code segments or neurons comprising an ANN. Typically, studies have been concerned with the replication of some of the most basic intelligent behaviours observed in the natural world, e.g. taxis and navigation. Elaborations on these experiments seek combinations or active switching of these modes to solve more complex tasks which approach usefulness in real world applications. In seeking to apply and test time delays in dynamic ANNs, in accordance with the research questions in Part I, these basic adaptive behaviours reported in Chapter II.1 provide a suitable starting point. In this way, this Chapter uses the models developed in Section II.4.2, including the work of Chapter II.5, in applying the methods of Chapter II.6.

III.2.1 T-Test Task

III.2.1.1 Hypothesis

The T-test task, inspired by the experiments of Jakobi [59], is introduced in detail in Chapter II.5 to demonstrate the evolution of explicitly encoded delays. Maze experiments such as this have formed the backbone of a con-

siderable body of work in both AI and the behavioural analysis of biological systems. As such it provides a useful link between the disciplines and, as a very commonly studied task with a range of known solutions, is an excellent candidate for inclusion here, as directed by the research questions of Part I.

A two-wheel robot modelled on the Khepera III (see Chapter II.4) is placed inside a T-shaped maze and must attempt to reach a goal indicated by a light stimulus presented at a certain point in the corridor. The fitness function scores individuals on how far they travel in the right direction and rewards them for avoiding contact with the walls and reaching the goal area in the shortest possible time.

Formally H_0, the null hypothesis is that including delays in the network does not increase the fitness in the T-test task. The alternative hypothesis H_1 is therefore that including delays increases the fitness.

III.2.1.2 Experimental Design

The experimental design is exactly as per Chapter II.5. Indeed, the results presented in detail there form a subset of the results presented herein for the range of time delay methods.

III.2.1.3 Results

The maximum fitness achieved throughout three evolutionary runs is presented in Table III.2.1 for each of the different delay types described in Chapter II.6. Each run consisted of 400 generations with an initial population of 1000 individuals.

These 12 data are then ranked from lowest to highest to obtain the ranks shown in Table III.2.2. These ranks are summed for the four groups to obtain $R_1 = 30$, $R_2 = 10$, $R_3 = 17$ and $R_4 = 21$. Also given in the table are the average ranks for each group, 10.0, 3.3, 5.7 and 7.0 respectively.

Now with these data, we may compute the value of simple form of KW

Run	Delay Type			
	None	(x,y)	Pattern	Direct
1	0.895702	0.827208	0.826459	0.895708
2	0.900830	0.825344	0.892837	0.740463
3	0.896826	0.759038	0.880125	0.900126

Table III.2.1: Maximum Normalised Fitness in Each Run

	Delay Type			
	None	(x,y)	Pattern	Direct
	8	5	4	9
	12	3	7	1
	10	2	6	11
R_j	30	10	17	21
$\bar{R}_j$	10.0	3.3	5.7	7.0

Table III.2.2: Ranked Results

as there are no tied values:

$$KW = \left[\frac{12}{N(N+1)} \sum_{j=1}^{k} n_j \bar{R}_j^2 \right] - 3(N+1) \tag{III.2.1}$$

$$KW = \left[\frac{12}{12(12+1)} [3(10.0)^2 + 3(3.3)^2 + 3(5.7)^2 + 3(7.0)^2] \right] - 3(12+1) \tag{III.2.2}$$

$$KW = 44.36 - 39 \tag{III.2.3}$$

$$= 5.36 \tag{III.2.4}$$

Since the observed value of KW (5.36) does not exceed 7.82 its probability of occurring under H_0 is greater than α, and so the hypothesis that delays do not improve the performance in this case is accepted.

Further to this conclusion, we wish to make a comparison between the various methods for determining the delays. If we take the differences between the average rankings $\bar{R}_j$ for each of the six possible comparisons, we have:

$$|\bar{R}_1 - \bar{R}_2| = |10.0 - 3.3| = 6.7 \tag{III.2.5}$$

$$|\bar{R}_1 - \bar{R}_3| = |10.0 - 5.7| = 4.3 \tag{III.2.6}$$

$$|\bar{R}_1 - \bar{R}_4| = |10.0 - 7.0| = 3.0 \tag{III.2.7}$$

$$|\bar{R}_2 - \bar{R}_3| = |3.3 - 5.7| = 2.3 \tag{III.2.8}$$

$$|\bar{R}_2 - \bar{R}_4| = |3.3 - 7.0| = 3.7 \tag{III.2.9}$$

$$|\bar{R}_3 - \bar{R}_4| = |5.7 - 7.0| = 1.3 \tag{III.2.10}$$

Here, $z_{a/k(k-1)} = z_{0.05/4(4-1)} = z_{0.0375} \approx 1.78$. The critical difference in

average rank is therefore given by:

$$z_{a/k(k-1)}\sqrt{\frac{N(N+1)}{12}\left(\frac{1}{n_u}+\frac{1}{n_v}\right)} = 1.78\sqrt{\frac{12(12+1)}{12}\left(\frac{1}{3}+\frac{1}{3}\right)} \quad \text{(III.2.11)}$$

$$= 1.78\sqrt{8.67} \quad \text{(III.2.12)}$$

$$= 5.24 \quad \text{(III.2.13)}$$

We can therefore conclude that the only populations which are statistically different for the given significance level of $\alpha = 0.05$ are the non-delayed and position modified, although a greater number of samples may provide further evidence that all populations are statistically similar.

Whilst considering the structures prerequisite in the evolved networks for their ability to solve the problem optimally, it is instructive to consider the genetic make-up of the populations and how it relates to their fitness. For each of the different delay methods the maximum fitness achieved over the course of each evolutionary run for each genome configuration (number of neurons and synapses) was averaged over all of the runs and analysed. This showed that a high fitness may be maintained for a significant number of additional links in the network, but added neurons cause a substantial falloff. Most significantly, all methods achieved their maximum fitness with a larger structure than the minimal genotype of the initial population. The average size of the optimal network was around 12 neurons and 20-30 synapses.

In summary, all of the methods achieved near optimal fitness and there was negligible difference in the efficacy of each method shown by the acceptance of the null hypothesis. The task has proven to be so easily solved that no method has an advantage and the size of the fittest networks is common to all. Delay methods demonstrated a lesser tolerance to larger network structures which may be due to the increased number of parameters encoded, but this did not hamper the search. More complex tasks must be sought to clarify the hypotheses stated previously.

III.2.2 Single Food Chemotaxis

III.2.2.1 Hypothesis

Chemotaxis is one of the most basic adaptive behaviours exhibited by organisms from unicellular Eukaryotes to *E.Coli* and beyond, as reported in Chapter II.1. It therefore serves as a useful benchmark in ER and Artificial Life (AL) and so supports the directive provided by the research questions of Part I. A simulated organism is provided with sensors to sample a diffused scent, produced by a number of items of food in the environment. The governing network may then control effectors (motors, thrusters or swimming appendages) to guide the organism to consume the food. With such a basic experiment for which optimal CTRNN solutions have been well proven, the goal is to define a similar baseline capability of delayed systems and to determine whether any improvements may be detected. The fitness of the individual is based on how quickly it can consume the available food, with a faster individual achieving a higher fitness.

Formally H_0, the null hypothesis is that including delays in the network does not increase the fitness for the Chemotaxis task. The alternative hypothesis H_1 is therefore that including delays increases the fitness.

III.2.2.2 Experimental Design

This experiment is modelled broadly on the work of Joachimzak and Wróbel in the evolution of GRNs for real time control of foraging behaviours [61]. The animat is modelled as a simulated Khepera III robot as per Section II.4.2, but with only two sensors and a bias input. In line with the desire to begin the evolutionary process with the simplest network possible, the networks were presented with two preprocessed sensor inputs $S1$ and $S2$

which represent the direction and distance of food sources (as per [61]).

$$S1 = \frac{1}{1 + e^{-\alpha(S_R - S_L)}} \tag{III.2.14}$$

$$S2 = \frac{2}{1 + e^{-\beta(S_R + S_L)}} - 1 \tag{III.2.15}$$

The sensors were arranged at angles of $\pm\pi/4$ on the body of the robot, $\alpha = 10.0$ and $\beta = 1.0$ were used to control the response of the functions.

The robot was randomly positioned within 200mm of the edges of an environment 1000mm $\times$ 1000mm, with $N_f = 10$ food items randomly positioned. No walls enclosed the environment and the robot was free to leave the boundaries but would not encounter any food. The scent perceived by the sensors S_L and S_R was proportional to the distance of the sensor from the food source as shown below:

$$s = \sum_F c_f \tag{III.2.16}$$

$$c_f = \begin{cases} (1 - d_f/D)^2 & \text{if } d \leq D; \\ 0 & \text{else.} \end{cases} \tag{III.2.17}$$

Where s is the sensor reading (S_L or S_R), c_f is the scent contribution from food f, d_f is the distance of food f from the sensor, F is the number of food items remaining in the environment and $D = 500mm$ is the scent diffusion radius. The resulting scent gradient for a sample initial food distribution can be seen in Figure III.2.1a. When the robot reaches a food item (the distance to the food is equal to the radius of the robot (65.81mm)), the food is removed from the environment and no longer contributes towards the sensor calculations and the energy of the robot is incremented. An interesting extension to this work would be to introduce a dynamic element into the simulation environment. This could take the form of food expiring and regenerating at intervals or a variety of other modes by which an additional level of complexity could be added.

The fitness of the robot is governed by the proportion of the available

Run	Delay Type			
	None	(x,y)	Pattern	Direct
1	0.759583	0.658542	0.670833	0.672292
2	0.734687	0.653958	0.681354	0.667604
3	0.730208	0.645729	0.681979	0.652812

Table III.2.3: Maximum Normalised Fitness in Each Run

food consumed and the time this took, so that concise directed behaviour had an evolutionary advantage over inefficient random walks which would eventually cover the whole area. Thus:

$$f = \frac{n_c(1 - \frac{1}{2}t)}{N_f T} \tag{III.2.18}$$

Where n_c is the number of food items consumed, t is the simulation time and T is the maximum simulation duration. In this way the maximum theoretical fitness per run was unity but the finite speed of the robot infers a practical limit of ≈ 0.7.

III.2.2.3 Results

The maximum fitness achieved throughout three evolutionary runs is presented in Table III.2.3 for each of the different delay types described in Chapter II.6. Each run consisted of 200 generations with an initial population of 1000 individuals and eight trials per individual per generation.

These 12 data are then ranked from lowest to highest to obtain the ranks shown in Table III.2.4. These ranks are summed for the four groups to obtain $R_1 = 33$, $R_2 = 8$, $R_3 = 23$ and $R_4 = 14$. Also given in the table are the average ranks for each group, 11.00, 2.67, 7.67 and 4.67 respectively.

Now with these data, we may compute the value of KW simply as there

	Delay Type			
	None	(x,y)	Pattern	Direct
	12	4	6	7
	11	3	8	5
	10	1	9	2
R_j	33	8	23	14
$\bar{R}_j$	11.00	2.67	7.67	4.67

Table III.2.4: Ranked Results

are no tied values:

$$KW = \left[\frac{12}{N(N+1)} \sum_{j=1}^{k} n_j \bar{R}_j^2 \right] - 3(N+1) \tag{III.2.19}$$

$$KW = \left[\frac{12}{12(12+1)} [3(11.0)^2 + 3(2.67)^2 + 3(7.67)^2 + 3(4.67)^2] \right] - 3(12+1) \tag{III.2.20}$$

$$KW = 48.15 - 39 \tag{III.2.21}$$

$$= 9.15 \tag{III.2.22}$$

Since the observed value of KW (9.15) exceeds 7.82, its probability of occurring under H_0 is less than α the performance of delayed and non-delayed systems is statistically different. In this case, the performance suffers a slight drop when delays are added to the system.

Further to this conclusion, we wish to make a comparison between the various methods for determining the delays. If we take the differences between the average rankings $\bar{R}_j$ for each of the six possible comparisons, we

have:

$$|\bar{R}_1 - \bar{R}_2| = |11.0 - 2.67| = 8.33 \quad \text{(III.2.23)}$$

$$|\bar{R}_1 - \bar{R}_3| = |11.0 - 7.67| = 3.33 \quad \text{(III.2.24)}$$

$$|\bar{R}_1 - \bar{R}_4| = |11.0 - 4.67| = 6.33 \quad \text{(III.2.25)}$$

$$|\bar{R}_2 - \bar{R}_3| = |2.67 - 7.67| = 5.00 \quad \text{(III.2.26)}$$

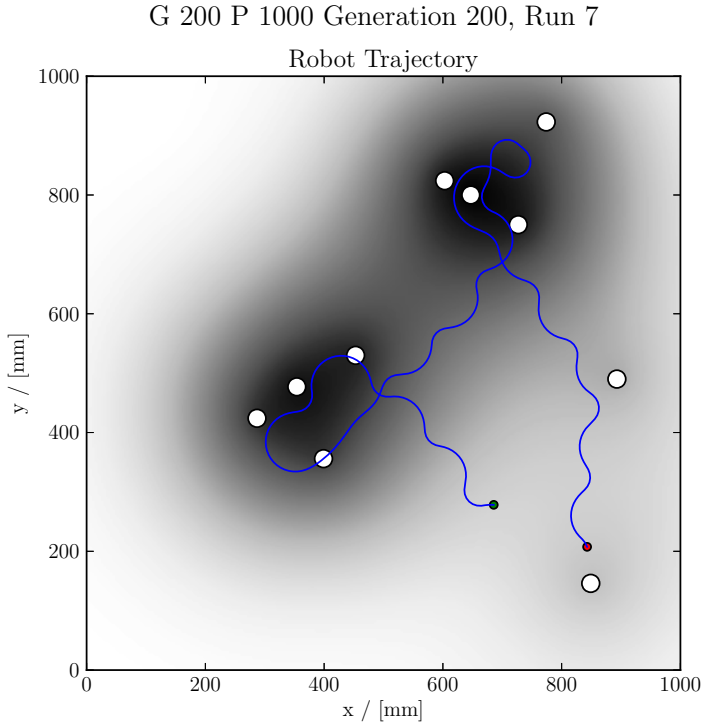
$$|\bar{R}_2 - \bar{R}_4| = |2.67 - 4.67| = 2.00 \quad \text{(III.2.27)}$$

$$|\bar{R}_3 - \bar{R}_4| = |7.67 - 4.67| = 3.00 \quad \text{(III.2.28)}$$

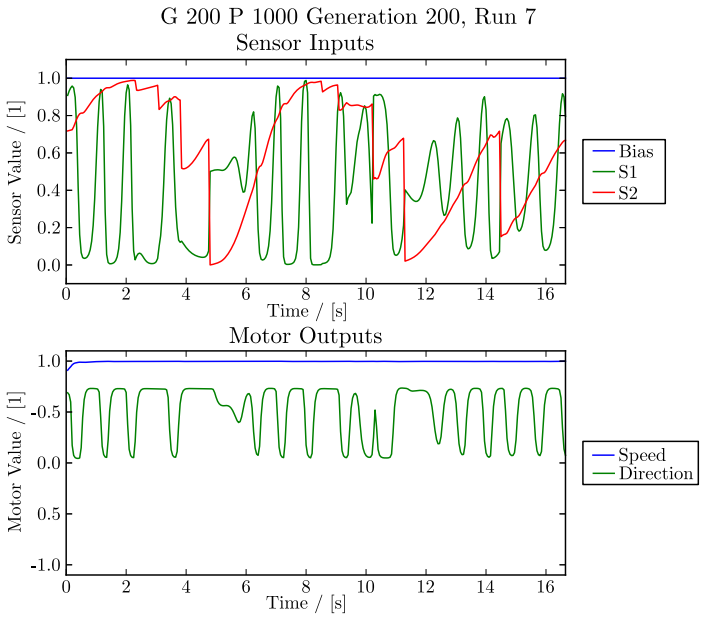
Here, the critical difference in average rank is the same as for Section III.2.1.3 (5.24) and we can therefore conclude that the no delay results are statistically different from the position and direct encoding for the given significance level of $\alpha = 0.05$. The pattern method however could not be distinguished at this confidence level.

Despite the slight difference in fitness, all methods were capable of achieving near optimal behaviour in the task. Figure III.2.1 shows results for the final generation of one pattern modified run. The robot starts at the green point and adopts a locally undulatory but globally direct path to the nearest high concentration of scent (Figure III.2.1a). After the majority of high density food items have been consumed the robot is still sensitive enough to locate the last remaining targets and complete the task in just over 16s. The constant changing of direction is a typical evolved solution which can be seen across all runs and methods. It allows the robot to better 'sniff out' the direction of highest scent gradient.

The sensor and motor time histories are presented in Figure III.2.1b. $S1$ can be seen to almost oscillate as the robot constantly changes direction back and forth its macro path. The distance measure $S2$ always rises, indicating a reduction in distance between the robot and the food, until the food is consumed and the robot experiences a step change in the input. The motors were running at full speed throughout the test.

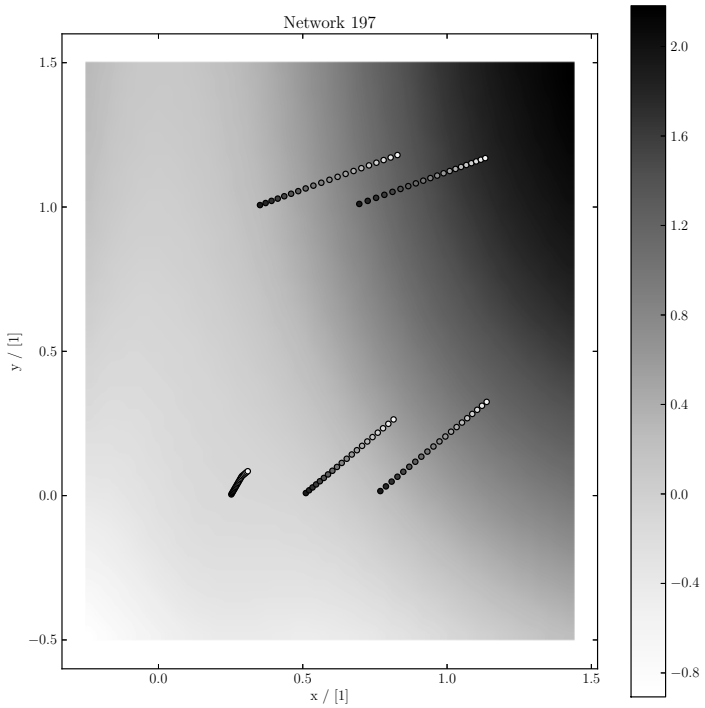


(a) Chemotaxis: Sampled Trajectory

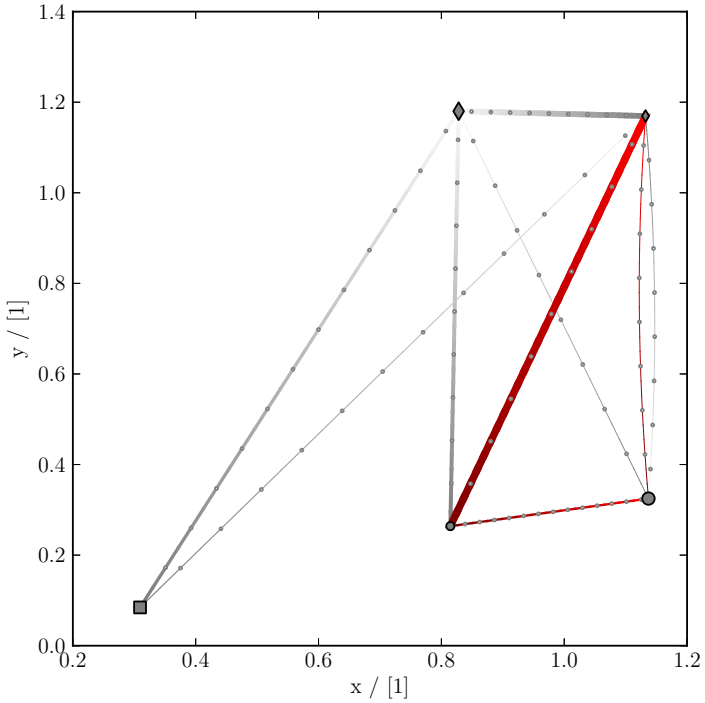


(b) Chemotaxis: Sample Sensor and Motor Time History

Figure III.2.1: Chemotaxis Sample Result



(c) Chemotaxis: Sampled Pattern Modified



(d) Chemotaxis: Sample Evolved Network

Figure III.2.1: Chemotaxis Sample Result (continued)

The evolved network which accomplishes this behaviour is shown in Figure III.2.1d, along with the pattern which modified its geometrical representation in Figure III.2.1c. The pattern is a simple 2D gradient which slightly compresses,the structure reducing the delays. Interconnections not present within the minimal genotype have been added through the course of the evolutionary process but there are no self-recurrent links.

The structure of the fitness function means that no robot can achieve a fitness of greater than 0.5 without collecting all the food items before the end of the simulation. The small decrease in fitness observed for delay methods likely corresponds to a reduction in overall speed throughout the tasks, due to slower information propagation through the network. This does not affect the ability of the network to solve the problem well, but does prevent the same levels of fitness being reached. Speed is often a convenient measure of fitness, but there are few occasions where it is vital. Particularly for more complex tasks, we might be pleased to sacrifice speed of response for an enhanced capability of control.

Analysis of the fitness distribution of evolved network structures as in Section III.2.1.3 showed a trend exhibited across all methods with a narrow plateau of high fitness solutions with a broadly linear relationship between the number of neurons and synapses (unsurprising given the nature of network structures). In contrast to the other examples given in this Chapter, no pronounced falloff was visible for larger structures. Most interestingly however is the fact that the no delay and direct methods both explored a larger genotype space. This illustrates that, whilst the position methods were capable of generating good solutions, the constraints of the geometrical representation have curtailed exploration of more complex solutions. This is not a problem for this simple task, but in a more complex scenario that is unsolvable until the minimal genotype has been somewhat elaborated, this may present a significant issue.

Chapter III.3

Robot Locomotion

As described in Section III.3 and Chapter II.1, legged robot locomotion has been one of the core avenues of ER research for many years, and continues to be so to this day. Robot gaits that are robust in complex and dynamic environments are hard to design manually and so, given the potential advantages for legged machines ability to cross difficult terrain, it is a great area of interest for the ER community. The research questions defined in Part I therefore compel us to consider robot locomotion alongside adaptive behaviour; indeed there is a substantial basis for this in the literature (see Chapter II.1). Therefore this Chapter presents the results of experiments into the application of delayed systems to walking robots, first considering a single leg and then an assembly of these limbs to form a more complex quadruped. The development of the tools to evaluate evolved gaits is undertaken in Chapter II.3 and, as per the overarching research questions, the various delay encoding techniques developed in Chapter II.6 are trialled for each experiment.

III.3.1 Single Leg Walker

III.3.1.1 Hypothesis

Chapter II.2 explores how even single node delayed systems can oscillate without external input to the network, in the manner of CPGs. We hypothesise that by adding delays and enhancing their behavioural dynamics, such delayed systems are better at generating oscillatory gaits for controlling a legged robot's locomotion. In the first instance we wish to explore a very simple system, a single leg, which should be easy to solve in comparison to the significantly more complex task of governing a quadruped gait. In addition, such a small initial governing network allows investigation into the hypothesis of Chapter II.2, that by adding connection delays CDRNNs may be smaller than previously possible to achieve the same functionality. Put simply, a higher fitness controller will move the body further in a given time period.

Stating this formally, H_0, the null hypothesis is that including delays in the network does not increase the fitness (ability to walk) of a single leg controller. The alternative hypothesis, H_1, is therefore that including delays increases the fitness.

III.3.1.2 Experimental Design

The approach to modelling a single leg with a sprawled posture as laid out in Section II.3.1 is used. This could be assembled into a system which is a 3D analogue of a 1D peg-leg walker or Chaplygin sled [52], where the leg is attached to a solid body constrained to move along an axis. This is similar to the 'rail roach' approach of Ghigliazza [40, 41] which was a first step towards a hexapod model and used single limb telescopic legs, rather than the three-axis leg considered here. A cartoon of the scheme is given in Figure III.3.1.

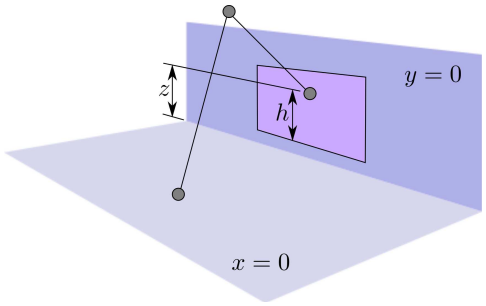


Figure III.3.1: Single Leg Walking Model

In this coordinate system, relative to the hip joint, the coordinates of the foot are:

$$x = \sin(\alpha)\left(l_1\cos(\beta) + l_2\cos(\gamma-\beta)\right) \quad \text{(III.3.1)}$$

$$y = \cos(\alpha)\left(l_1\cos(\beta) + l_2\cos(\gamma-\beta)\right) \quad \text{(III.3.2)}$$

$$z = l_1\sin(\beta) - l_2\sin(\gamma-\beta) \quad \text{(III.3.3)}$$

Realistic limits are set on the range of motion of the leg as follows:

$$-\frac{\pi}{4} \leq \alpha \leq \frac{\pi}{4} \quad \text{(III.3.4)}$$

$$-\frac{\pi}{4} \leq \beta \leq \frac{\pi}{4} \quad \text{(III.3.5)}$$

$$0 \leq \gamma \leq \frac{3}{4}\pi \quad \text{(III.3.6)}$$

A highly simplified model would consider a single leg attached to a massless body constrained to move solely on the y-plane. The leg is attached at a certain distance above the ground, and the body will rest on the ground unless the leg lifts it upwards. When in this elevated phase of the gait, the foot is assumed to be fixed in its x-position and is allowed to slide along the ground plane in the y-direction, as any frictional forces would be reacted by the y-plane body constraint. In this way, the leg can raise the body off the ground, swing through an angle, moving the body forwards in the x-direction, and then rest the body on the ground whilst the leg resets its position. This massless model ignores dynamic friction and inertial effects,

but is illustrative of real world behaviour.

A three node fully connected network where each node directly controls the α, β or γ leg angle directly is the smallest network capable of controlling all degrees of freedom in the model and thus is used as the minimal genotype for the evolutionary runs. Single neuron outputs were mapped to leg angles as follows:

$$\alpha = \frac{\pi}{4}(-1 + 2o_1) \tag{III.3.7}$$

$$\beta = \frac{\pi}{4}(-1 + 2o_2) \tag{III.3.8}$$

$$\gamma = \frac{3\pi}{4}o_3 \tag{III.3.9}$$

Motors or other actuators controlling each degree of freedom in the leg system have a finite response speed to any given demand. If the output of the network changes faster than this, the leg actuation will lag behind. Speed of response is typically quoted under no-load conditions and will often slow considerably under load, depending on the specifics of the actuators. This is simply modelled by limiting the speed of movement to a maximum of 60°/s (or $\pi/3$ radians/s). This prevents the evolution of infeasible systems which oscillate as quickly as possible with an infinitesimal amplitude. In combination with the equations above, these expressions completely define the mapping from node output to foot movement. In the simulation, the leg lengths (l_1 and l_2) were both 0.1m, and the initial height (h) was 0.05m.

The fitness of each controller is simply the distance that body is moved in the simulation time. For the evolutionary mechanism we must define a maximum achievable fitness from which the error for each individual is calculated at the end of each generation. In this case, the maximum achievable fitness can be calculated using trigonometry and expressed in a simplified form as below:

$$f_{max} = \frac{2T\pi\sqrt{(l_1 + l_2)^2 - h^2}\sin(\pi/4)}{s} \tag{III.3.10}$$

Where T is the simulation duration, l_1 and l_2 are the limb lengths, h is the body height and s is the speed limit in radians per second. This maximum value is used to normalise the fitness scores achieved. It should be noted that this score is an absolute maximum, based on the geometry of the leg and the maximum speed of movement. It does not take into account the need to decelerate and reverse direction at the end of each swing and various other factors which cause the maximum fitness of any realisable gait to fall somewhat below this theoretical value. Results presented are in terms of this normalised fitness unless otherwise stated.

III.3.1.3 Results

The maximum fitness achieved in the final generation of five evolutionary runs is presented in Table III.3.1 for each of the different delay types described in Chapter II.6. Each run consisted of 200 generations with an initial population of 1000 individuals. The task under investigation is entirely deterministic, in the way that with no noise added to the system and constant starting conditions across all simulations the results for a particular network configuration are exactly repeatable, and therefore only a single trial run per individual per generation is required. This determinism was also manifest in a rapid rise from initially poor fitness scores to a stable maximum fitness over the first 50-100 generations, depending on the run. This trait was observed across all of the runs. What follows is a statistical analysis of this data with further discussion and analysis of detail regarding the structure of evolved solutions and their behaviour.

These 20 data are then ranked from lowest to highest, with tied values bearing the average rank of the group, to obtain the ranks shown in Table III.3.2. These ranks are summed for the four groups to obtain $R_1 = 15$, $R_2 = 67$, $R_3 = 52$ and $R_4 = 76$, as shown in Table III.3.2. Also given in the table are the average ranks for each group, 3.0, 13.4, 10.4 and 15.2

Run	Delay Type			
	None	(x,y)	Pattern	Direct
1	0.016667	0.120365	0.119130	0.120616
2	0.016667	0.120795	0.116318	0.119786
3	0.016667	0.108460	0.118181	0.116552
4	0.016667	0.119414	0.117337	0.119648
5	0.016667	0.112427	0.112341	0.119240

Table III.3.1: Maximum Final Generation Normalised Fitness

	Delay Type			
	None	(x,y)	Pattern	Direct
	3	18	13	19
	3	20	9	17
	3	6	12	10
	3	15	11	16
	3	8	7	14
R_j	15	67	52	76
$\bar{R}_j$	3.0	13.4	10.4	15.2

Table III.3.2: Ranked Results

respectively.

Now with these data, we may compute the value of KW (taking into account the single group of tied values for the non-delayed group):

$$KW = \frac{\left[\frac{12}{N(N+1)}\sum_{j=1}^{k} n_j \bar{R}_j^2\right] - 3(N+1)}{1 - \frac{\left[\sum_{i=1}^{g}(t_i^3 - t_i)\right]}{(N^3 - N)}} \qquad \text{(III.3.11)}$$

$$KW = \frac{\left[\frac{12}{20(20+1)}[5(3.0)^2 + 5(13.4)^2 + 5(10.4)^2 + 5(15.2)^2]\right] - 3(20+1)}{1 - \frac{[(5^3-5)]}{(20^3-20)}} \tag{III.3.12}$$

$$KW = \frac{75.39 - 63}{0.98} \tag{III.3.13}$$

$$= 12.58 \tag{III.3.14}$$

Since the observed value of KW (12.58) exceeds 7.82, its probability of occurring under H_0 is less than α and so the hypothesis that delays do not improve the performance in this case is rejected.

Further to this result, we wish to make a comparison between the various methods for determining the delays. If we take the differences between the average rankings $\bar{R}_j$ for each of the six possible comparisons, we have:

$$|\bar{R}_1 - \bar{R}_2| = |3.0 - 13.4| = 10.4 \tag{III.3.15}$$

$$|\bar{R}_1 - \bar{R}_3| = |3.0 - 10.4| = 7.4 \tag{III.3.16}$$

$$|\bar{R}_1 - \bar{R}_4| = |3.0 - 15.2| = 12.2 \tag{III.3.17}$$

$$|\bar{R}_2 - \bar{R}_3| = |13.4 - 10.4| = 3.0 \tag{III.3.18}$$

$$|\bar{R}_2 - \bar{R}_4| = |13.4 - 15.2| = 1.8 \tag{III.3.19}$$

$$|\bar{R}_3 - \bar{R}_4| = |10.4 - 15.2| = 4.8 \tag{III.3.20}$$

Here, $z_{a/k(k-1)} = z_{0.05/4(4-1)} = z_{0.0375} \approx 1.78$. The critical difference in average rank is therefore given by:

$$z_{a/k(k-1)}\sqrt{\frac{N(N+1)}{12}\left(\frac{1}{n_u} + \frac{1}{n_v}\right)} = 1.78\sqrt{\frac{20(20+1)}{12}\left(\frac{1}{5} + \frac{1}{5}\right)} \tag{III.3.21}$$

$$= 1.78\sqrt{14} \tag{III.3.22}$$

$$= 6.66 \tag{III.3.23}$$

We can therefore conclude that all the delayed systems achieve a significantly higher fitness score compared to the non-delayed, but that no differ-

ence can be detected within the different methods for determining delays for the given significance level of $\alpha = 0.05$.

Figure III.3.2 presents a sample result from one of the five runs where the time delays are directly controlled by the evolutionary process. These results are typical of high fitness individuals and serve to illustrate the kind of solution reached. Figure III.3.2a shows how the individual accrued fitness over the duration of the simulation through repeated cycles where the foot was in contact with the ground and moved the body forwards at a fairly constant speed. Periodic plateaus are readily apparent where the foot lifted off the ground as it returned to begin the cycle again. The network which governed this behaviour is shown in Figure III.3.2b. Positive weights are shown in black and negative weights in red, with the gradient indicating the direction of information flow and the thickness corresponding to the strength of the connection. The three principle nodes governing the control of the leg angles are at each point of the triangle, with the size of the node proportional to its time constant. Each information store along each synapse is marked with a grey point with the total per synapse corresponding to the number of timesteps by which the propagating information was delayed. Key features to observe are additional intermediate node and synapses added during the course of evolution, most notably the self-recurrent synapse which typically generates much of the oscillatory behaviour.

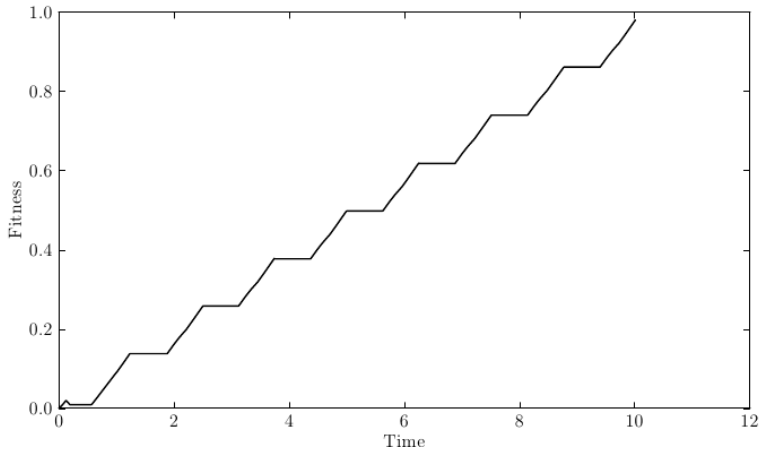
The output of the neurons is shown in Figure III.3.2c. It can be seen that the trajectory is initially undulatory, as the initial conditions stored in each synapse propagate through and affect the dynamics, but quickly approaches a stable orbit. This effects the movement of the foot shown in Figure III.3.2d where an oscillatory cycle exists, raising and lowering the foot into and out of contact with the ground, propelling it forwards.

This behaviour is observed in all of the delayed systems. The non-delayed systems, by contrast, are unable to oscillate within the limitations of this

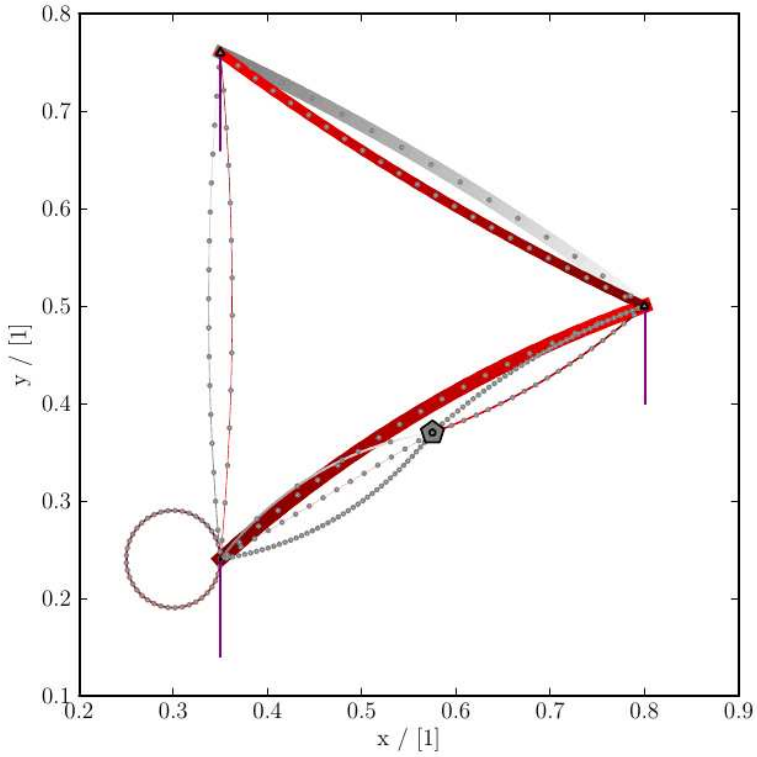
experiment, even with the addition of recurrent synapses and additional genetic elements as evident in the successful (delayed) systems. This limited their fitness to a single ground stroke, moving the body forwards to the limit of the leg's range of movement. The result of this is a very defined, but low, maximum fitness which was easy to achieve and which is evident in the lack of variability between trials of the non-delayed system. Whilst CTRNNs are well known for their ability to oscillate, this showcases the enhanced ability for small delayed networks to demonstrate this behaviour in a manner controllable and exploitable by an evolutionary process.

Analysis of the genome structure for evolved solutions, as per Section III.2.1.3, was undertaken and the results presented in Figure III.3.3. With this information we can make some judgements as to the suitability of the minimal genotype selected for this experiment. Common features to all of the surfaces plotted are the linear relationship between the number of neurons and the number of synapses. This was to be expected for a number of reasons. The minimal genotype was fully connected and so an increase in synapses beyond self-recurrent connections required the addition of more neurons. Also, the inclusion of further nodes in the network necessarily split existing synapses into two, further increasing their number.

The results of this analysis for the non-delayed networks shown in Figure III.3.3a is telling, as there is a very defined maximum fitness and this is achievable over a wide range of genome configurations. Even large numbers of additional neurons and synapses were capable of reaching this peak. In contrast to this, the picture is very different when we consider the case where the delays are encoded by controlling the position of nodes in the network directly, see Figure III.3.3b. Here, peak fitness is achieved with networks only slightly larger than the minimal genotype with a marked falloff to a minimal fitness plateau for larger networks. The peak is observed within 3-5 neurons and 6-20 synapses.

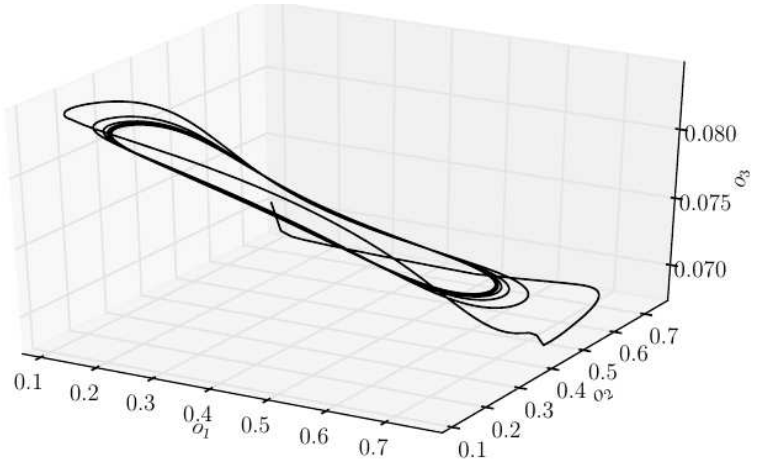


(a) Fitness Time History

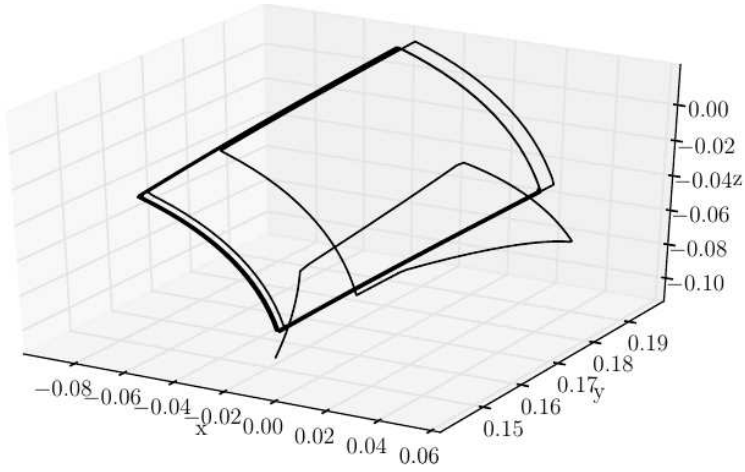


(b) Best Evolved Network

Figure III.3.2: Single Leg Task Sample Result

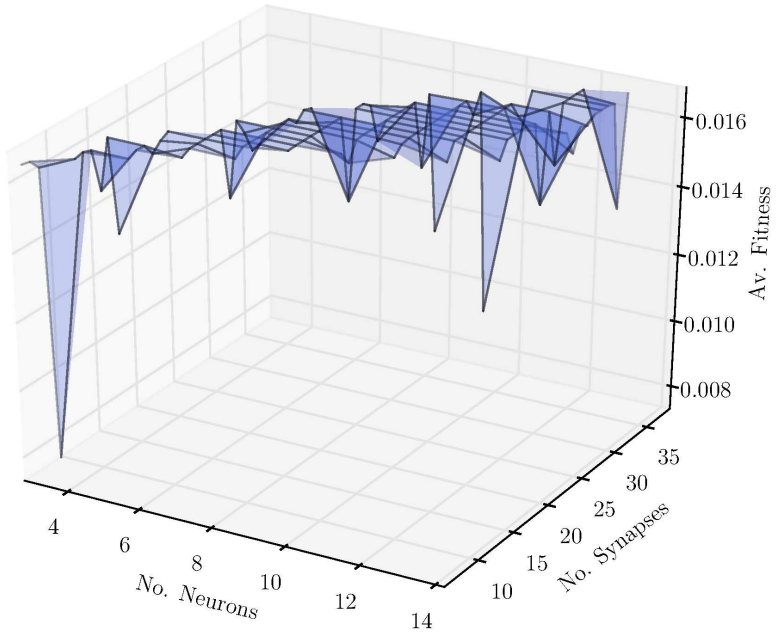


(c) Neuron Outputs

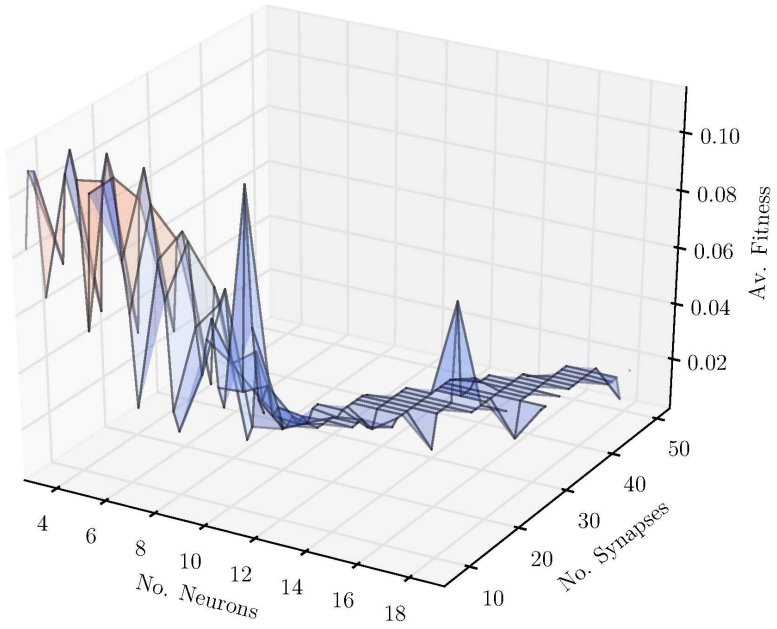


(d) Foot Movement

Figure III.3.2: Single Leg Task Sample Result (continued)

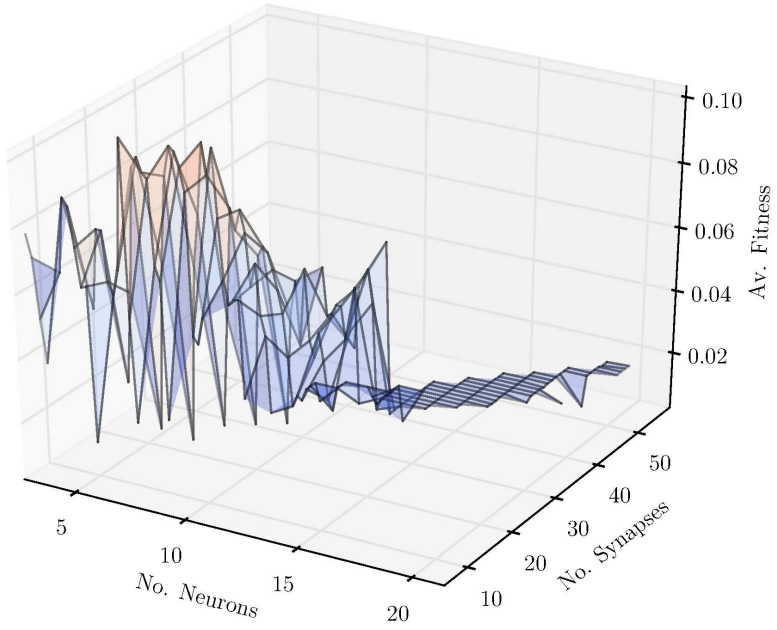


(a) None

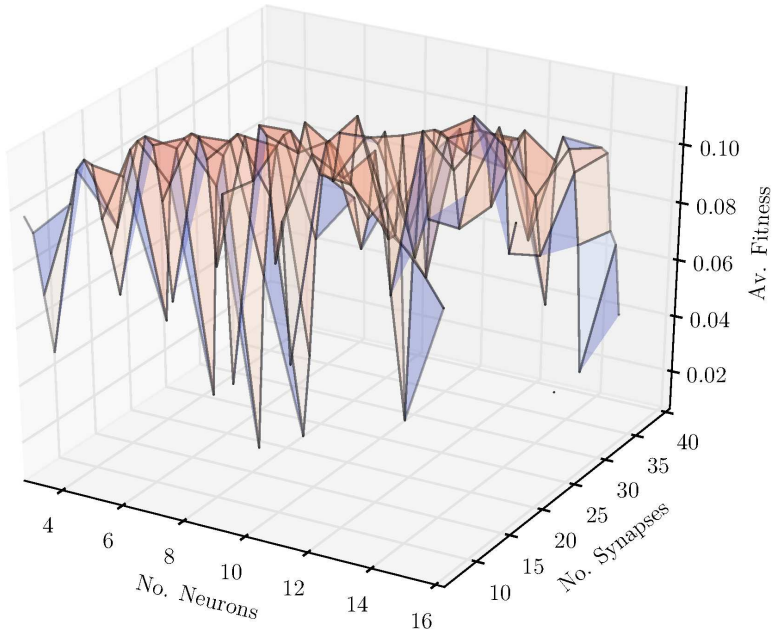


(b) Position

Figure III.3.3: Single Leg Task Genome Analysis



(c) Pattern



(d) Direct

Figure III.3.3: Single Leg Task Genome Analysis (continued)

When the position of the network nodes, and therefore the delays, was encoded using a pattern network, as in Figure III.3.3c, the fitness peak was enlarged and biased slightly in favour of increased number of synapses. In the above case, the minimal genotype of 3 neurons and 6 synapses had near optimal performance with only a slight increase due to further complexity, but this is more marked here. The minimal structure only achieved around 80% of the average population maximum, with the peak observed at around 5 neurons and 20 synapses. Whilst smaller, there is still an appreciable plateau of minimal fitness which was explored during evolution but in which no good solutions were found. The surface shows considerably more noise than the other cases because this visualisation ignores the effect of the pattern network which completes each individual's genome.

Figure III.3.3d shows the case where the delays were encoded directly, independent of the spatial configuration of the network. There is no such marked fitness drop-off as the size of the network increases. Whilst the optimum is still around 6 neurons and 20 synapses, good solutions were found when the network was double this size with only a small degradation in fitness.

From this, it is clear that the spatial encoding of the delays in the network causes a marked fall in the ability of evolution to find fit solutions for larger networks. It seems clear that this is due to the constraints applied by such a representation, as it becomes complex or impossible to find configurations of node positions which generate good behaviour. This could be considered in both a positive and negative light, in that in the comparison of performance between both spatial methods and the direct encoding there was no clear difference in performance. It is possible that the biologically inspired spatial constraint is helping to limit the search to within an optimal area (the peak fitness structure was roughly the same, independent of the method for finding the delays). However, whilst this may be true it is likely that for more

complex tasks which may require more complex networks to solve them this manner of representation may prevent high fitness solutions being found. As discussed in Chapter II.6, the extension of the spatial representation into 3D increases the number of possible delay configurations and thus aids evolutionary searches for larger networks.

What is clear however, is that the evolutionary process has been observed to correctly explore solutions of increased complexity which exceed that of the peak fitnesses observed. We can therefore be optimistic that the parameters of the evolutionary search correctly matched that of the experimental design and that at no point was the process constrained from reaching regions of higher fitness.

III.3.2 Quadruped Walking

III.3.2.1 Hypothesis

As stated in Section II.3.1, a natural extension of the single leg system considered in Section III.3.1 is to assemble multiple legs together to simply model a wide range of biological legged creatures. The bulk of Chapter II.3 is concerned with the development of a computationally minimal environment suitable for the evaluation of evolved gaits within a GA. Quadrupeds have been widely studied in ER, including recent work on the application of HyperNEAT by Clune et al. [30], see Section II.1.5. This makes them highly relevant for inclusion here when considering the overarching research question of Part I to apply delayed systems to commonly studied tasks in ER for benchmarking and comparison.

For the evolution of a stable gait it is necessary to coordinate the movement of all limbs of the robot, so that it moves and remains either statically or dynamically stable. A manually designed gait was evaluated in Chapter II.3 and it has been hypothesised that a CDRNN system would provide a

performance enhancement over a simpler CTRNN model. Similarly to the single leg experiment above, the fitness of an individual is simply the distance that the body is propelled in a given time. A better controller will move the robot further.

Therefore, the null hypothesis H_0 is that including delays in the network does not increase the fitness (ability to walk) of a quadruped robot controller. The alternative hypothesis H_1 is that including delays does increase the fitness.

Experimental Design

The simulated robot was identical to that developed in Section II.4.1 and used in Chapter II.3 and similarly the success (fitness) of the robot was determined by the straight line distance travelled from the origin in the ground plane over the duration of the simulation ($f = \sqrt{x^2 + y^2}$). The robot was initially given a neutral stance of $\alpha = 0.0$, $\beta = \pi/8$ and $\gamma = \pi/2$ so that it was stably supported on all four legs with the body raised off the ground. The leg motors had a maximum speed limit of π radians per second and each leg was controlled by two motor neurons. One controls the swing of the leg (α) and the other controls whether the leg is raised or lowered (β and γ). For the four legs of the robot there are therefore eight motor neurons. This is a slight simplification over the single leg experiment, but it reduces the size of the network structure. The desired motor angles at each stage were determined as below:

$$\alpha_l = \frac{\pi}{4}\left(-1 + 2o_{l,1}\right) \tag{III.3.24}$$

$$\beta_l = \frac{\pi}{4}\left(-1 + 2o_{l,2}\right) \tag{III.3.25}$$

$$\gamma_l = \frac{\pi}{8}\left(4 + 3o_{l,2}\right) \tag{III.3.26}$$

Where α_l, β_l and γ_l are the joint angles for leg l, and $o_{l,1}$ and $o_{l,2}$ are the two motor neurons for leg l. If the desired change in leg angles per time

step exceeds the speed limit of the motors, then the angles are moved at maximum speed.

As the experiment in Section III.3.1 proved that systems without delays could not demonstrate the required oscillatory behaviour this experiment was designed to avoid a trivial replication of this result. Along with a constant bias node, there is one other input to the network which is a square wave varying from -1 to +1 with a period of four seconds. This enabled systems without delay to use the driving oscillations and shifted the focus to how leg oscillations could best be coordinated in time to generate a stable gait.

The simulation and leg movements (interpolated between motor neuron outputs) were updated every 0.0002s with network updates synchronously every 0.01s. Should the robot become inverted the simulation is ended. The maximum fitness of the robot was taken to be the same as for the single leg case as defined in Equation III.3.10, with the fitness of each individual simply being the Cartesian distance from the origin at the end of the simulation.

III.3.2.2 Results

The maximum fitness achieved throughout three evolutionary runs is presented in Table III.3.3 for each of the different delay types described in Chapter II.6. Each run consisted of 50 generations with an initial population of 100 individuals and one trial per individual per generation as the simulation is deterministic. As the system was provided with an oscillatory input, the scope of the evolutionary search is really optimisation rather than synthesis and so should be solvable using smaller populations over fewer generations than in the other tasks. Even with the attempt at performance optimisation through simulation development in Chapter II.3 this is still the most computationally expensive simulation reported in this Thesis.

These 12 data are then ranked from lowest to highest to obtain the

Run	Delay Type			
	None	(x,y)	Pattern	Direct
1	0.488852	0.516227	0.516227	0.576120
2	0.576191	0.561164	0.596107	0.597205
3	0.510423	0.651434	1.838552	0.423061

Table III.3.3: Maximum Normalised Fitness in Each Run

	Delay Type			
	None	(x,y)	Pattern	Direct
	2	4	4	7
	8	6	9	10
	3	11	12	1
R_j	13	21	25	18
$\bar{R}_j$	4.33	7.00	8.33	6.00

Table III.3.4: Ranked Results

ranks shown in Table III.3.4. These ranks are summed for the four groups to obtain $R_1 = 13$, $R_2 = 21$, $R_3 = 25$ and $R_4 = 18$. Also given in the table are the average ranks for each group, 4.33, 7.00, 8.33 and 6.00 respectively.

Now with these data, we may compute the value of KW simply as there are no tied values:

$$KW = \left[\frac{12}{N(N+1)} \sum_{j=1}^{k} n_j \bar{R}_j^2 \right] - 3(N+1) \tag{III.3.27}$$

$$KW = \left[\frac{12}{12(12+1)} [3(4.33)^2 + 3(7.00)^2 + 3(8.33)^2 + 3(6.00)^2] \right] - 3(12+1) \tag{III.3.28}$$

$$KW = 40.97 - 39 \quad \text{(III.3.29)}$$

$$= 1.97 \quad \text{(III.3.30)}$$

Since the observed value of KW (1.97) does not exceed 7.82 its probability of occurring under H_0 is greater than α, and so the hypothesis that delays do not improve the performance in this case is accepted.

Further to this conclusion, we wish to make a comparison between the various methods for determining the delays. If we take the differences between the average rankings $\bar{R}_j$ for each of the six possible comparisons, we have:

$$|\bar{R}_1 - \bar{R}_2| = |4.33 - 7.00| = 2.67 \quad \text{(III.3.31)}$$

$$|\bar{R}_1 - \bar{R}_3| = |4.33 - 8.33| = 4.00 \quad \text{(III.3.32)}$$

$$|\bar{R}_1 - \bar{R}_4| = |4.33 - 6.00| = 1.67 \quad \text{(III.3.33)}$$

$$|\bar{R}_2 - \bar{R}_3| = |7.00 - 8.33| = 1.33 \quad \text{(III.3.34)}$$

$$|\bar{R}_2 - \bar{R}_4| = |7.00 - 6.00| = 1.00 \quad \text{(III.3.35)}$$

$$|\bar{R}_3 - \bar{R}_4| = |8.33 - 6.00| = 2.33 \quad \text{(III.3.36)}$$

Here, the critical difference in average rank is the same as for Section III.2.1.3 (5.24) and we can therefore conclude that none of the encoding methods are statistically different for the given significance level of $\alpha = 0.05$.

No method for encoding time delays is demonstrably better than a standard CTRNN model by the measure of the simple fitness measure employed in this experiment. Even with the relatively small population and number of generations all runs successfully generated some form of locomotion.

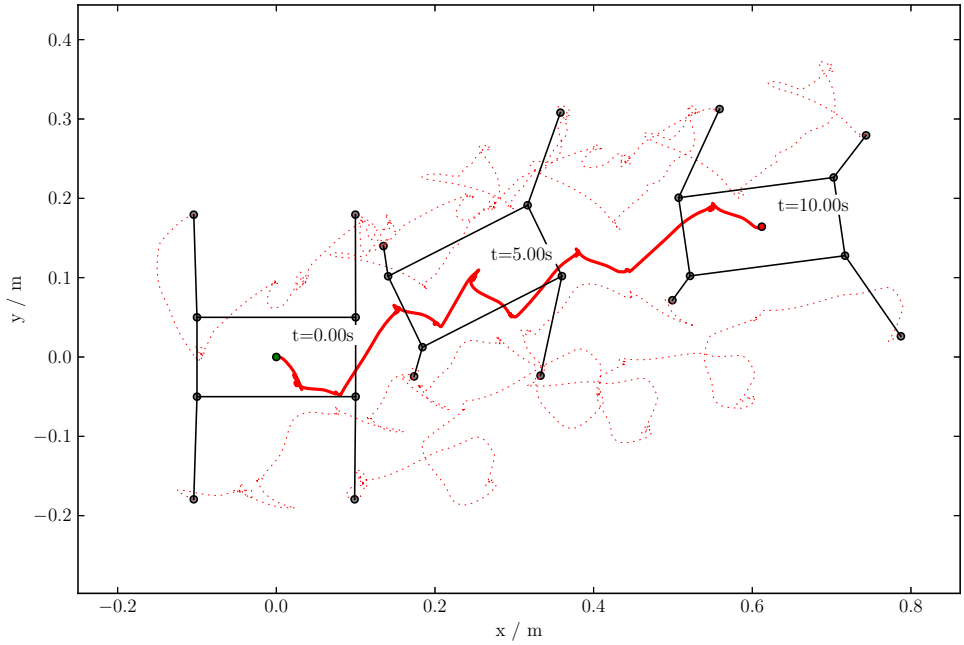
Given that the maximum fitness of unity would require instantaneous transition from maximum speed one way to maximum speed the other with totally splayed legs the fitnesses observed were very high. There were three distinct strategies adopted across all runs which were qualitatively more or less successful despite achieving comparable fitness scores.

Figure III.3.4 shows an example result of a final generation solution from a directly encoded delay network. Figure III.3.4a and Figure III.3.4b show the top and side views respectively of the simulated trajectory. An exaggerated zig-zag pattern can be seen as the robot moves forwards with a relatively flat body throughout. The predicted torques are shown in Figure III.3.4c which are in-line with those developed during the manually designed gain in Chapter II.3 with occasional transitory loads of up to 0.8 Nm. This gait uses all four legs to propel the robot fairly straightly in the positive x-direction. As can be seen from the torque diagram the network has scheduled leg oscillations to form a stable (1,4,3,2) quadruped crawling gait which was the goal of the exercise.

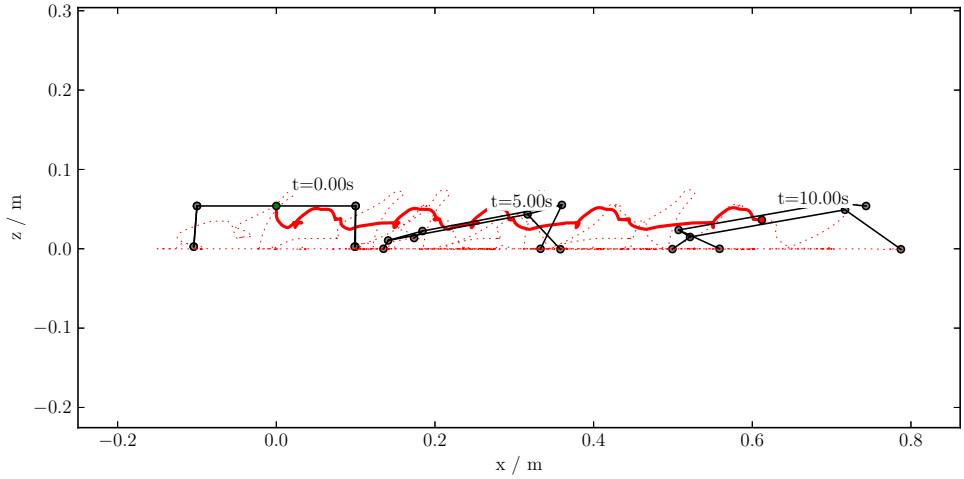
Systems without delays did not appear capable of scheduling four leg oscillations but instead relied on a tripod gait to move the body forwards. This was successful in the manner that it generated statistically similar fitness scores to the sample result shown but are less suited to real world environments.

A third notable strategy was demonstrated in the final run of the pattern encoded delay network which achieved a fitness significantly above that of the theoretical maximum. This violation is not too significant as the justification of the maximum fitness calculation was predicated on the robot moving forwards. This gait scuttled crab-wise which combined both the β and γ leg angles to enable swifter leg movement, and therefore a greater maximum fitness, than the α leg angle based calculation. Whilst the fitness achieved in this manner was high the gait was not robust.

Both delayed and non-delayed dynamic systems have been shown to be able to generate a locomotory gait satisfactorily with network inputs as described above. Whilst there was no statistically significant difference between their fitness scores in a more qualitative manner the delayed systems were shown to be capable of a more realistic quadruped gait. Whilst mea-

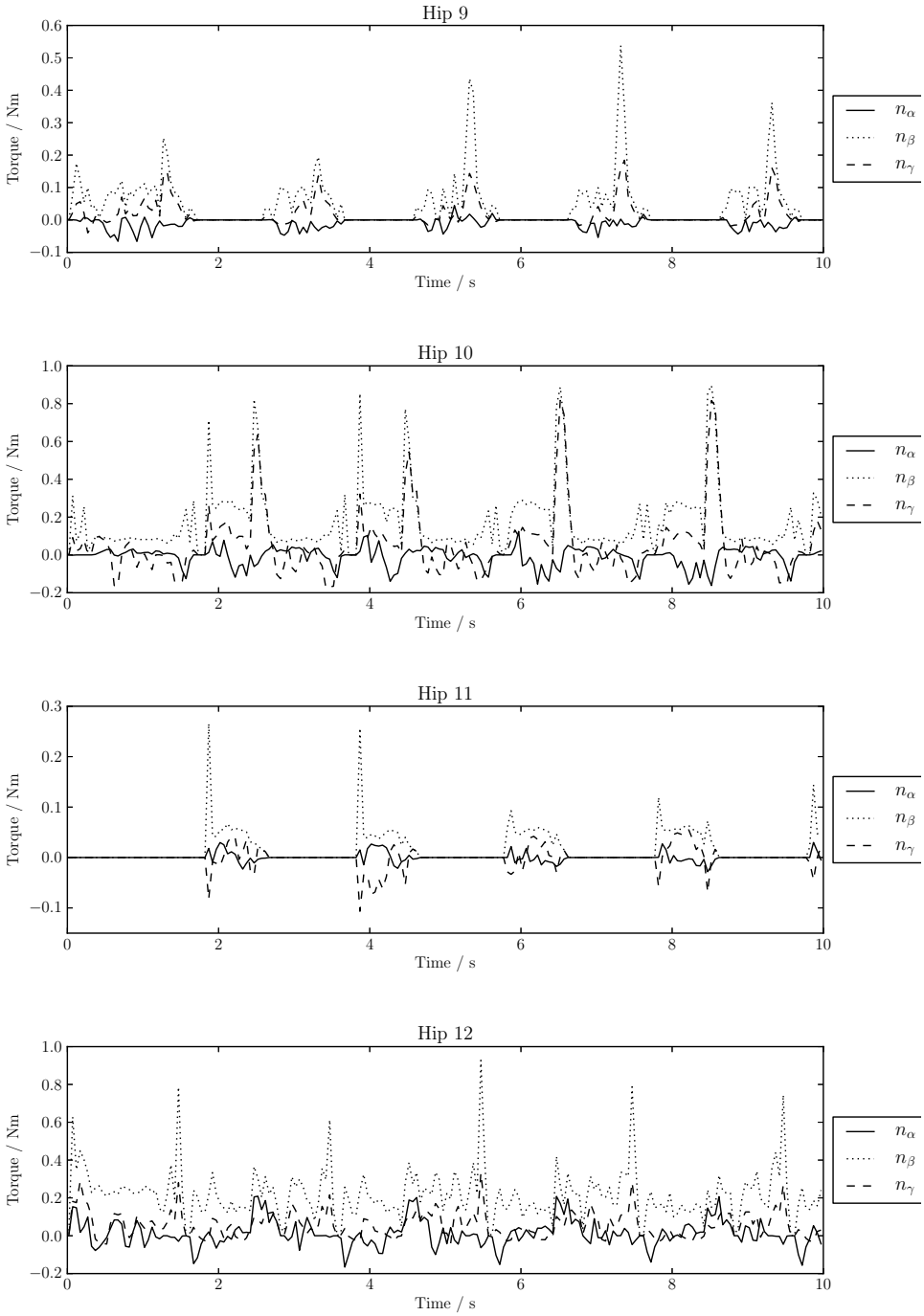


(a) Top View



(b) Side View

Figure III.3.4: Quadruped Walking Example Trajectory



(c) Leg Torques

Figure III.3.4: Quadruped Walking Example Trajectory (continued)

sures such as the average height and flatness of the body, the contact time for all legs and the degree of leg movement could be included in a more elaborate fitness measure designed to reward those behaviours we see the be better in some qualitative manner. This is not trivial however and even if a more complex fitness function avoids the 'bootstrapping problem' the results may well be otherwise than desired, and so often simplest is best. That said it would be interesting to further explore this experiment without an oscillatory input and with some form of incremental evolution for a larger population over a greater number of generations.

Chapter III.4

Conclusions

In this Part experiments across a wide range of commonly studied tasks in ER were reported with the aim of proving if time delays enhanced performance, if smaller solution were possible and how the spatial encoding of delays compared to direct encoding. What emerged was a mixed picture, which will briefly be summarised here.

Investigation into the evolution of gaits for a simple single leg system without sensor input demonstrated a massive boost in fitness when delays were allowed into the system. Without them, a CTRNN network of that size simply was not able to develop the oscillations required to drive the gait.

Extending this work to the 3D simulation of quadruped gaits, as developed in Chapter II.3, all network types were capable of developing locomotion with a square-wave oscillatory input the network to avoid a trivial replication of the single leg results. Whilst this was true, it was only the delayed systems that demonstrated the ability to schedule a quadruped gait effectively with all non-delayed solutions relying on tripod gaits.

Given the combined results of the single leg and quadruped walking experiments it seems that delayed dynamic systems are well suited to the control task with the facility to distribute responses in time to coordinate effector movement and to generate self-generating oscillatory network ele-

ments from a single node. There are many and varied potential avenues of exploration available which are beyond the scope of this Thesis, but could be fascinating subjects of future study.

Other examples in adaptive behaviour have been less positive. Delays have either had no effect, or slightly decreased the achievable maximum fitness of the system (without seriously harming the ability to solve the problem). This may largely be put down to the choice of fitness measures, but it exemplifies that there is a cost associated with the increase in complexity which may be unjustified if there is no clear benefit as there was to the single leg system.

What is required, and what has thus far eluded discovery, is a sufficiently complex, but still solvable, task where the delays may present an advantage. Without this we can simply conclude that delays have been shown to make a significant improvement to legged locomotion applications, and may be confidently applied to other domains which require the same characteristic. Allowing for much simpler systems to solve these problems is a significant advantage, but this has not yet been proven in the search for intelligent adaptive behaviour.

The conclusions that can be drawn as to the efficacy of the different methods for determining delays are less clear. Certainly the spatial encoding of delays did not cause a drop in fitness over the direct method. In one case (the chemotaxis example) the pattern encoding method was the only delay case which was indistinguishable from the non-delay case at that significance level ($\alpha = 0.05$). However, these techniques clearly restricted the search to lower complexity solutions or led to a significant falloff as the network structure became more complex. This would suggest that for anything other than a simple genotype, the direct method for encoding delays is likely most appropriate.

Ultimately, it has been demonstrated that delayed dynamic neural sys-

tems are at least as capable as traditional CTRNNs for the range of experiments reported here. The additional gamut of dynamic behaviour available upon the introduction of delays, as explored in Chapter II.2, directly enabled a significant boost in fitness for the three-node single leg walker. By adapting the synaptic delays to coordinate limb movement delayed systems enabled a more sophisticated gait to evolve than was seen in systems without delays. In these cases there is a direct link between additional dynamics introduced and a reasonable hypothesis for improved performance. In the experiments in adaptive behaviour, where there is not such a direct link, no such discernible improvement was witnessed. Whilst we hypothesise that the ability of such delayed networks to generate pattern generating nodes which may be switched on or off and combined in an enormous variety of ways may be useful in ER this was not borne out by the experiments undertaken here. Whether the experiments simply did not require such structures for success or the fitness functions did not reward or encourage such solutions is a matter for future study. With the success in locomotion tasks and the open potential for experiments in adaptive behaviour delayed dynamic systems for evolved control remain an interesting and promising field in complex systems research.

Part IV

Summary & Conclusions

Summary & Conclusions

This Thesis proposes the inclusion of connection time delays within the modelling of recurrent dynamic networks in the search for intelligent control architectures. The effect this has on the dynamics of the systems has been comprehensively determined, and a scheme for introducing them into an GA developed. A range of methods for determining the time delays are proposed and their efficacy in developing fit solutions across a range of typical ER tasks evaluated. In this Chapter the various conclusions contained in this Thesis are brought together and the achievements and weak points are presented alongside potential topics for future investigation.

In Chapter II.2 it was shown how adding connection transport delays to RNNs enables single neuron oscillation without external input, in the manner of a CPG, and smaller circuits to express complex switched oscillatory behaviour than previously possible. This behaviour is fundamentally different from the traditional CTRNN model and the enhanced capability could be harnessed to generate behaviour and control architectures previously impossible.

The research questions in Part I encourage the application of these systems to ER tasks in adaptive behaviour and legged robot locomotion. To support investigation of the latter a novel computationally efficient, minimalistic approach to a dynamic simulation environment was successfully developed and validated using a real robot.

To obtain realistic and realisable results in ER it is essential 'close the loop' between hardware and software, as it is through the interaction of an embodied controller with the environment that behaviour is emergent. In Chapter II.4 a simple, low cost quadruped robot was developed to validate the simulation from Chapter II.3 and 2D wheeled robot models, using hardware derived parameters, were developed to support the experiments from

Chapters II.5 and III.2. A common code architecture was developed to guarantee compliance and equivalence between the PyPy accelerated software for genetic optimisation and real-time Python control.

Time delayed RNNs were applied for the first time, to the author's knowledge, to a robotic control task in Chapter II.5 commonly studied in the field of ER in accordance with the research questions. Evolved controllers successfully solved the T-junction task and it was shown through sensitivity analysis that the solution was highly dependent on the configuration of delays and that removing the delays in the system reduced its ability to solve the task.

To study CDRNNs within the context of ER it is necessary to encode the delay values within a genome and subsequently map it to a phenotype. The experiments in Chapter II.5 adopted a simple method by which the real valued delays are encoded in the genome directly. However, the research question asks how else we might encode these delays and what effect this might have on the efficiency or capability of evolved solutions. In Chapter II.6 a range of methods are presented based upon the novel, biologically inspired concept of spatial representation of the network and determination of delays based upon connection lengths. Such methods were shown to be potentially more efficient (requiring fewer additional parameters than the direct method) but equally might constrain the complexity of possible solutions. These hypotheses were tested throughout Part III through application of each method to a range of commonly studied tasks in adaptive behaviour and robot locomotion and comparison to the behaviour of unmodified CTRNNs.

The enhanced dynamic behaviour discovered in Chapter II.2 demonstrably enabled a three neuron network without external input to establish an efficient single leg tetrapod gait which was impossible without delays. Spatial encoding methods resulted in a marked fitness falloff as the number of

neurons and synapses increased which may prevent high fitness solutions being developed for more complex tasks. When extended to a quadruped gait, using the simulation of Chapter II.3, there no quantitative fitness difference between the delayed and non-delayed systems. However, only the delayed systems were observed to correctly schedule the leg oscillations to form a stable quadruped crawling gait with the others relying on simple tripod gaits to move the body forwards.

Within the experiments reported on adaptive behaviour in Chapter III.2 there was no supporting evidence that time delays enhanced performance or allowed for smaller solutions than previously possible. Whilst there was no metabolic cost with evaluating larger networks and thus no evolutionary pressure to generate the smallest possible solutions, analysis of the evolved genomes typically demonstrated an increase in fitness once the initial minimal genotype was elaborated. Therefore it should have been possible to validate the hypothesis should the data have supported it.

Thus we have shown that adding time delays into CTRNNs increases their dynamic capability which may be harnessed to achieve behaviour previously not possible, particularly when applied to legged robot locomotion. This Thesis has considered a number of methods by which the delays can be governed by an EA but for the majority of cases a simple direct encoding is likely most suitable as no restriction is placed on the possible configuration of delays. Whilst legged robot control applications were demonstrably well suited to this control architecture, no such observation was made with reference to commonly studied tasks in adaptive behaviour. Whilst the dynamic capabilities of the enhanced networks are clear, and one can certainly anticipate their utility in building complex and powerful neural circuits, the experiments chosen did not require such features for optimal solutions to be found.

Achievements

For the first time synaptic time delays have been introduced into the CTRNN model which has been the mainstream neural network model of choice in ER for decades. Time delays have previously only been used in recurrent neural network models for the preconditioning of sensor data. In contrast to this, here they are advocated for the improved development of adaptive behaviour and legged robot gaits. It was hypothesised that the increased dynamics of the system may enable behaviours previously unattainable or less complex solutions for established problems.

The dynamics of the modified form of CTRNN, the CDRNN, was investigated using a single self-recurrent node. Previous studies of DDE systems have tended to focus on the stability rather than the dynamics which is the focus of this research. It is the dynamic responses which are different from the established model and may be harnessed for increased performance. Even a single delayed node has been shown to be capable of acting as a pattern generator, which is not possible in the equivalent CTRNN. This capability may greatly enhance larger systems, as such pattern features may be expressed or suppressed at any point in the network without explicit stimuli, and in much smaller systems than previously possible.

With clear potential benefits to the evolutionary search for legged robot gaits a minimalistic 3D simulation was developed in the spirit of Jakobi for the computationally efficient evaluation of evolved behaviour. As the number of limbs on a robot decreases so does the inherent stability of the design. For a quadruped it is necessary to consider 3D dynamic stability in evaluating gaits, whereas early work in hexapod locomotion was typically 1D in nature. Modern robotics simulations typically interface with mature physics engines, such as Open Dynamics Engine (ODE), but this can place significant constraints on the researcher. In the case of this research the EA used was written in Python and accelerated using PyPy, which requires

pure Python modules and cannot integrate with standard rigid body simulation packages. A simulation was designed which simplified the assembly of the robot into a single body to which forces are applied from legs driven by trigonometric relationships. Impulse based simulation techniques were applied to allow single step solution in parallel with the neural network evaluation. Simulation results for a manually designed gait were compared to a physical robot developed for the purpose of the simulation validation and a high degree of accuracy was observed. Crucially, the predicted distance travelled was almost identical between simulation and reality which is the main method by which evolved solutions are evaluated.

A number of novel methods for encoding time delays into the genome for artificial evolution were developed. Most significantly, methods of spatial representation of the network where the time delays are determined by the geometrical length of the connections between nodes were developed. A co-evolved CPPN was harnessed to place a gradient pattern under evolutionary control which in turn modified the position of network nodes through a hill climbing algorithm. These methods of representation were analysed and are expected to be more efficient than direct delay encoding for larger networks, but also significantly restrict the configuration of delays achievable.

Using these methods time delay networks were applied to a a range of tasks commonly studied in ER, from adaptive behaviour to legged robot locomotion. The pattern generating ability of delayed nodes enabled good solutions for single leg systems where it was not possible before. No fitness difference between methods was discernible, demonstrating that for small networks the restriction of values that delays could take had no effect. However, it is equally true that for such a small network the efficiency gains for position encoding are small, and pattern encoding is likely worse.

Weak Points

The overarching issue with respect to the evolutionary search for adaptive and intelligent behaviour is its failure to achieve good results to realistic problems, despite initial promising progress with simplified models [45]. This issue is thus far common to all concepts proposed in the search for true AI. This is perhaps one of the most complex and devoutly sought topics of research in all of human endeavour and it is perhaps not surprising that there are a myriad of complex issues. Briefly then, those most applicable to this research must be discussed and then more specific weaknesses presented in context.

Evolutionary methods are, for all their strengths, subject to a number of constraints. Chiefly among these is the computational cost of simulating large populations over a substantial number of generations. Biological evolution had the luxury of very large populations and an awfully long time (unless one considers micro-systems of bacteria which may evolve on much more rapid time scales e.g. in the human gut). The computational capacity at our disposal has increased dramatically since the early days of ER, but still falls well short of that we might anticipate will be required to investigate truly complex systems.

Central to evolutionary methods is the requirement to rank individuals in the population based upon some measure of their 'fitness'. The majority of experiments in ER use an explicit fitness function represented mathematically and based on quantifiable measures in the system. This causes difficulties not only when it is difficult to characterise success in this manner, but also when perceptibly simple fitness measures lead to undesirable solutions through the Law of Unintended Consequences. This can make it extremely difficult, and a subject of significant trial and error, to arrive at a fitness function which behaves as the human experimenter desires. Often a task may appear too complex for early generations, resulting in a uniformly

poor score and the evolutionary search downgrading to a random sampling (the Bootstrap Problem). From the beginning a number of proposals have been made to resolve this such as competitive co-evolution or supervised evolution. These present their own problems in the way that tasks must be restructured to fit the framework, or in the practical population and generation limits for human supervised evolution.

More specifically, the failure to prove the hypothesis suggesting that delays may increase performance and reduce network sizes for adaptive behaviour tasks may be said to be a failure to find the right task to investigate. Whilst the tasks were selected to establish the performance of delay enhanced networks within the context of established ER literature a more complex or tailored task would be more likely to highlight any performance gains. There is an inherent difficulty in finding the limits of complexity which can reasonably be solved using current computational techniques. Even 'intelligent' search algorithms such as GAs require a significant number of parameters which greatly effect the search, i.e. how broad it is or how long it should run for. Whilst there is evidence that the search spaces of GAs are not as vast as one might expect, it is still a complex problem. However, with that said complex time delay systems are prevalent in the natural and engineered worlds. The challenge is therefore to find examples with sufficient complexity and advantages for a time delayed system but which are still solvable.

Further Work

Should it be warranted, there are a number of avenues for future consideration as extensions to the research presented in this Thesis. It has been shown that for simple problems in the field of adaptive behaviour CDRNNs have thus far failed to demonstrate any improvement in capability. Whilst the extended dynamics are clear, the true validation of this work would

be to design an experiment where these dynamics are desirable. In Chapter II.2 a simple example of how a small CDRNN system could behave as a self-switched pattern generator was used as an illustration of the complex behaviour possible. Unsurprisingly the main avenue for further work is therefore in the application of the techniques developed to more adventurous examples.

The spatial representation of network delays described here uses a linear model to relate changes in connection length to the delay values. As mentioned in Chapter II.6 a simple extension would be to consider a range of non-linear mappings which would allow for the exploration of a wider range of delay values or allow for finer tuning.

The analysis of possible configurations for 2D spatial representations highlighted the constraints the encoding imposed. Chapter II.6 also proposed the extension of the network geometry into 3D in order to allow for a greater range of connection configurations. This would add a corresponding increase in evolutionary parameters, but would potentially be worth investigating given the evidence presented above that position encoding may slightly hamper the evolution of optimal solutions whilst not rendering them impossible.

Real robots have been used throughout this Thesis as inspiration and for validation of the methods proposed. Given the promising results for the application of CDRNN systems to robot gaits, a fascinating line of enquiry might be application to evolution in hardware of real legged robots. In the opposite direction the techniques established for deriving dynamic systems with substantial delays could as well be applied to the modelling of biological systems such as complex eukaryotes.

Summary

In this Thesis the concept of including connection delays into RNN models for enhanced adaptive behaviour and robot gaits has been explored. A simple modification to the standard CTRNN enables a complex suite of dynamic behaviour which may be harnessed in our search for intelligent systems. These dynamics have been shown to be efficacious in the evolution of gaits but, despite potentially useful mechanisms clearly available, no perceptible advantage for experiments in adaptive behaviour was detected in the examples used.

Of the various methods proposed for encoding network time delays the spatial representation method was shown to be more efficient for larger networks, but tended to restrict the range of possible solutions and could thus impair fitness. For anything other than a simple genotype a direct representation of time delays in the system is the most viable.

The majority of techniques or advances in science and AI present select advantages to tailored cases. The work presented in this Thesis suggests the potential for useful application to real world systems of robot gaits but failed to demonstrate tangible improvements to the evolution of adaptive behaviour. In replicating previous works to establish a proven record of performance, we have seemed to eliminate the necessary complexity which might vindicate these methods. The much used form of the CTRNN has been very popular for its power and flexibility and it would require something further by way of experimentation to capture the broader dynamics introduced in this work. These extended dynamics are clear and one can easily construct toy circuits which might be generated and assembled by evolutionary methods to solve complex behavioural problems. The ultimate summary of this Thesis must therefore be one of promise, as yet largely unfulfilled, but presenting a fascinating facet of complex systems research for future exploration.

Part V

Appendices

Appendix V.A

Numerical Integration

At the core of investigating dynamic systems is the numerical integration of differential equations. There are a vast array of methods for acheiving this, with a whole range of benefits nad disadvantages. Some are tailored for specific purposes, but the majority of Evolutionary Robotics literature uses the simplest methods of all. In this Appendix two methods are outlined for completeness. They are somewhat crude and inaccurate for anything other than very small timesteps. However, the simplicity of their implementation is a boon to those applications where we are considering mixed systems of asynchronous updating dynamic networks and physical simulations.

V.A.1 Euler Method

This is the simplest possible approximation to the solution of a first order differential equation, but it can be used for any order ordinary differential equations.

$$\dot{f} \approx \frac{f^{t+\Delta t} - f^t}{\Delta t} \tag{V.A.1}$$

$$f^{t+\Delta t} = f^t + \Delta t \dot{f} \tag{V.A.2}$$

Here, accuracy is sacrificed for simplicity and is largely determined by the step size.

V.A.2 Verlet Integration

The Verlet integration method uses the first three terms of the Taylor expansion to obtain an approximation to the integral. This method is valuable to us, as it is more accurate than the Euler method but still only requires knowledge of the differentials at the current time step.

$$f(t + \Delta t) \approx f(t) + \Delta t f'(t) + \frac{1}{2}\Delta t^2 f''(t) \tag{V.A.3}$$

A variant of this procedure is known as Velocity Verlet integration which manipulates the above expression to remove the requirement for knowledge of the first derivative term.

$$f(t + \Delta t) \approx 2f(t) - f(t - \Delta t) + \frac{1}{2}\Delta t^2 f''(t) \tag{V.A.4}$$

This expression is undefined at $t = 0$, so the standard Verlet integration method is used for the first step only.

V.A.3 Fourth Order Runge-Kutta Method

Whilst the majority of ER literature relies on the Euler method for integrating dynamic neural network systems the standard numerical integration scheme for phyiscal simulations is the Fourth Order Runge-Kutta (RK4) method which uses the weighted average of derivatives at invervals during each timestep. The scheme has local truncation error on the order of $\mathcal{O}(h^5)$ and total accumulated error is on the order $\mathcal{O}(h^4)$.

If $\dot{y} = f(t, y)$ and $y(t_0) = y_0$ then for successive timesteps (of interval h) from the initial time t_0 and corresponding initial condition y_0 the next value of y is given by:

$$y_{n+1} = y_n + \frac{1}{6}h\left(k_1 + 2k_2 + 2k_3 + k_4\right) \tag{V.A.5}$$

$$t_{n+1} = t_n + h \tag{V.A.6}$$

For $n \in \mathbb{N}$, where:

$$k_1 = f(t_n, y_n), \tag{V.A.7}$$

$$k_2 = f(t_n + \frac{1}{2}h, y_n + \frac{h}{2}k_1), \tag{V.A.8}$$

$$k_3 = f(t_n + \frac{1}{2}h, y_n + \frac{h}{2}k_2), \tag{V.A.9}$$

$$k_4 = f(t_n + h, y_n + hk_3), \tag{V.A.10}$$

Appendix V.B

Recap of Selected Vector and Matrix Mathematics

V.B.1 Skew-symmetric Matrices

A skew-symmetric matrix is defined as follows. If $\mathbf{a} = [a_x, a_y, a_z]^T$, then:

$$\tilde{\mathbf{a}} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{V.B.1}$$

From this we can see that $\tilde{\mathbf{a}}^T = -\tilde{\mathbf{a}}$.

V.B.2 Vector Dot Product

The dot product (.)of two vectors is defined as follows:

$$\mathbf{a}.\mathbf{b} = \sum_{i=1}^{n} a_i b_i = |\mathbf{a}||\mathbf{b}| \cos\theta \tag{V.B.2}$$

Where θ is the angle between the two vectors.

V.B.3 Vector Cross Product

The cross product ($\times$) of two vectors may be calculated in a number of ways, but most suitable for our needs is by using a skew matrix (See (V.B.1)).

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} \tag{V.B.3}$$

Appendix V.C

General Rotations and Translations

To solve the EOM for three-dimensional dynamic simulation we must be able to relate the coordinates of the feet of the robot from the body fixed reference frame (x', y', z') to the ground fixed reference frame (x, y, z). If the vector $\mathbf{s}'^P$ locates a point P in the x', y', z' frame, then in x, y, z this vector is $\mathbf{As}'^{\mathbf{P}}$. When the body origin is at coordinates $\mathbf{r}$ in x, y, z. The transformed vector is:

$$\mathbf{r}^P = \mathbf{r} + \mathbf{A}\mathbf{s}'^P \tag{V.C.1}$$

Similarly, if the body coordinate system is rotating as well as translating, then the velocity of a point P which is stationary in the x, y, z frame is given by:

$$\dot{\mathbf{r}}^P = \dot{\mathbf{r}} + \dot{\mathbf{A}}\mathbf{s}'^P \tag{V.C.2}$$

Where:

$$\dot{\mathbf{A}} = \tilde{\boldsymbol{\omega}}\mathbf{A} = \mathbf{A}\tilde{\boldsymbol{\omega}}' \tag{V.C.3}$$

If the point P is moving with velocity $\dot{\mathbf{s}}'^P$ this vector may simply be transformed into the global frame and superposed with the velocity found in

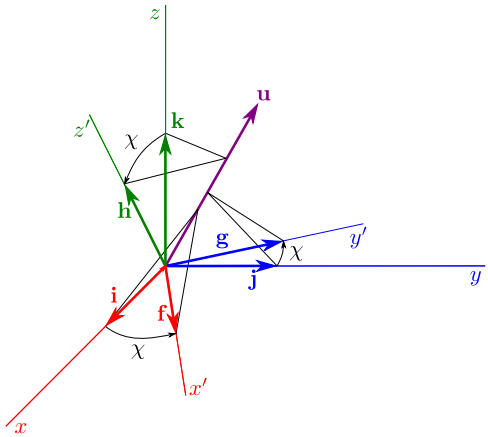


Figure V.C.1: Euler's Theorem

Equation V.C.2, to get:

$$\dot{\mathbf{r}}^P = \dot{\mathbf{r}} + \dot{\mathbf{A}}\mathbf{s}'^P + \mathbf{A}\dot{\mathbf{s}}'^P \tag{V.C.4}$$

The rotation matrix $\mathbf{A}$ is best represented through the use of Euler parameters. Euler's Theorem states that any rotation in 3D between two coincident orthogonal coordinate systems may be defined by a single rotation (χ, positive counter clockwise) about an axis (defined by unit vector $\mathbf{u}$), shown in Figure V.C.1. Then:

$$e_0 = \cos\frac{\chi}{2} \tag{V.C.5}$$

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \mathbf{u}\sin\frac{\chi}{2} \tag{V.C.6}$$

$$\mathbf{p} = [e_0, e_1, e_2, e_3]^T \tag{V.C.7}$$

These must obey the *Euler Parameter normalization constraint* where $\mathbf{p}^T\mathbf{p} = \mathbf{e}_0^2 + \mathbf{e}_1^2 + \mathbf{e}_2^2 + \mathbf{e}_3^2 = 1$. The rotation matrix is then given by:

$$\mathbf{A} = 2\begin{bmatrix} e_0^2 + e_1^2 - \frac{1}{2} & e_1e_2 - e_0e_3 & e_1e_3 + e_0e_2 \\ e_1e_2 + e_0e_3 & e_0^2 + e_2^2 - \frac{1}{2} & e_2e_3 - e_0e_1 \\ e_1e_3 - e_0e_2 & e_2e_3 + e_0e_1 & e_0^2 + e_3^2 - \frac{1}{2} \end{bmatrix} \tag{V.C.8}$$

The inverse of the rotation matrix is simply it's transpose, so:

$$\mathbf{A}^{-1} = \mathbf{A}^T \tag{V.C.9}$$

This allows us to easily transform vectors between the body reference frame and the ground reference frame. Inspection of the EOM reveals that the forces must be applied in the ground frame and the torques applied in the body frame. This requires us to be able to perform the inverse transform of that given in Equation V.C.1. Given a known vector $\mathbf{s}$ and using Equation V.C.9, we can calculate $\mathbf{s}$:

$$\mathbf{s}' = \mathbf{A}^T\mathbf{s} \tag{V.C.10}$$

We will know the coordinates of the feet in the body reference frame directly in order to apply the sliding and reaction forces, but it would be easy to calculate them from a global position with the location and orientation of the body reference frame.

An additional matrix $\mathbf{G}$ is required to find $\dot{\mathbf{p}}$ and are made up of the Euler Parameters which is intimately related to the rotational matrix $\mathbf{A}$.

$$\mathbf{G} = \begin{bmatrix} -e_1 & e_0 & e_3 & -e_2 \\ -e_2 & -e_3 & e_0 & e_1 \\ -e_3 & e_2 & -e_1 & e_0 \end{bmatrix} \tag{V.C.11}$$

Appendix V.D

Mass and Inertia of a Uniform Body

The mass and inertial characteristics of a uniform body, of the type illustrated in Figure V.D.1, with mass density Γ are:

$$m = \Gamma abc \tag{V.D.1}$$

$$J_{x'x'} = \frac{1}{12} m(b^2 + c^2) \tag{V.D.2}$$

$$J_{y'y'} = \frac{1}{12} m(a^2 + c^2) \tag{V.D.3}$$

$$J_{z'z'} = \frac{1}{12} m(a^2 + b^2) \tag{V.D.4}$$

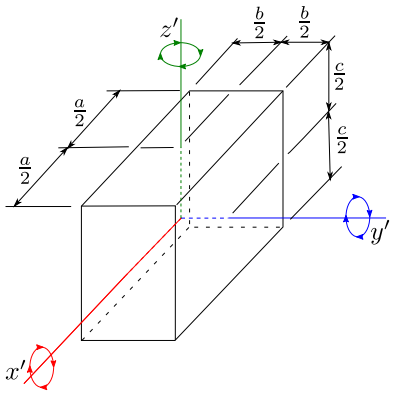


Figure V.D.1: Rectangular Prism, reproduced from [48]

Using the following formations:

$$\mathbf{J}' = \begin{bmatrix} J_{x'x'} & 0 & 0 \\ 0 & J_{y'y'} & 0 \\ 0 & 0 & J_{z'z'} \end{bmatrix} \tag{V.D.5}$$

As $\mathbf{J}'$ is diagonal, finding its inverse is considerably simplified as we can use the identity:

$$\text{diag}(a_1, a_2, ..., a_n)^{-1} = \text{diag}(a_1^{-1}, a_2^{-1}, ..., a_n^{-1}) \tag{V.D.6}$$

To transform this into the global frame, we simply calculate:

$$\mathbf{J}^{-1} = \mathbf{A}\mathbf{J}'^{-1}\mathbf{A}^T \tag{V.D.7}$$

Glossary

1D

One-Dimensional. 102, 105, 221, 250

2D

Two-Dimensional. 135, 202, 219, 247, 254

3D

Three-Dimensional. 29, 103, 105, 107, 117, 133, 135, 194, 221, 243, 250, 254, 263

a priori

knowledge that is known independently of experience. 37, 50

AI

Artificial Intelligence. 23, 208, 252, 255

AL

Artificial Life. 212

animat

a contraction of anima-materials; an artificial animal including physical robots and virtual simulations. 31, 212

ANN

a mathematical model or computational model that tries to simulate the structure and/or functional aspects of biological neural networks. 25, 27, 34, 39, 41–43, 46, 65, 75, 177, 189, 207

autonomy

the capacity of a system to make a decision about its actions without the involvement of another system or operator. 56

BEC

Battery Elimination Circuit. 128, 139

biomimetic

mimicking biology. 199

CDRNN

Continuous Delayed Recurrent Neural Network. 74, 75, 78, 80, 86–89, 96, 158, 174, 175, 189, 191, 221, 234, 248, 250, 253, 254

CE

Cellular Encoding. 39

chemotaxis

The ability to move in an orientation with respect to a source of a chemical. 31, 47, 67, 203, 244

COM

Centre of Mass. 108, 115, 128, 131

CPG

Central Pattern Generator. 67, 68, 70, 221, 247

CPPN

Compositional Pattern Producing Network. 39, 187, 189, 191–194, 197, 198, 251

CTRNN

Continuous Time Recurrent Neural Network. 2, 12, 24, 25, 28, 37, 43, 52, 63–70, 72–81, 87, 90, 93, 95, 96, 99, 101, 102, 148, 157, 167, 171, 174, 175, 179, 191, 192, 202–205, 212, 228, 235, 243, 245, 247–250, 255

DDE

Delay Differential Equation. 24, 28, 73, 74, 76, 77, 99, 102, 250

DGRN

Dynamic Recurrent Gene Network. 49

DH

Denavit-Hartenberg. 106

DNA

Deoxyribonucleic Acid. 47, 48

DOF

Degree of Freedom. 68, 109

EA

Evolutionary Algorithm. 25, 29, 202, 249, 250

ECG

Echo CardioGram. 46

embryogenesis

the formation and development of an embryo. 49

EOM

Equations of Motion. 106, 108, 109, 115, 121, 123, 264

epigenetic

A term referring to the non-genetic causes of a phenotype. 47

ER

Evolutionary Robotics. 2, 25, 27–29, 31, 34, 36–38, 42, 43, 51, 53, 58, 67, 69–71, 96, 99, 101–103, 120, 135, 149, 155, 157, 159, 162, 175, 176, 178, 205, 207, 212, 220, 234, 243, 245, 247, 248, 250–253, 258

eukaryote

an organism whose cells contain a nucleus and other structures (organelles) enclosed within membranes. 254

exteroceptor

any receptor that responds to stimuli outside the body. 56, 58

FIFO

First-In-First-Out. 99

FSM

Finite State Machine. 40

GA

a search heuristic that is based on biological evolution. 28, 37, 38, 42, 59, 63, 234, 247, 253

genotype

the genetic composition of an organism. 39, 67, 69

GOFAI

Good Old Fashioned Artificial Intelligence. 24, 41

GP

Genetic Programming. 39

GRN

a collection of DNA segments in a cell which interact with each other. 2, 24, 47, 48, 50, 51, 54, 69, 70, 157, 212

HyperNEAT

A method for evolving CPPNs based on the NEAT technique.. 68, 69, 189, 234

interoceptor

any receptor that responds to stimuli inside the body. 56, 58

IR

Infra-Red. 13, 46, 74, 147, 148, 155, 159, 161, 162

JIT

Just-In-Time. 149

LCP

Linear Complementary Problem. 112, 123, 124

MLP

Mult-Layer Perceptron. 74

morphology

the form, structure and configuration of an organism. 36, 55, 68

NEAT

NeuroEvolution of Augmenting Topologies. 39, 69, 158, 159, 164, 187, 189, 192, 196, 199

neuroethology

the evolutionary and comparative approach to the study of animal behavior and its underlying mechanistic control by the nervous system. 45

Ni-Cd

Nickel-Cadmium. 128

ODE

Open Dynamics Engine. 250

ODE

Ordinary Differential Equation. 24, 47, 48, 68, 74, 124

photoreceptor

a specialized neuron able to detect, and react to light. 56, 57

phototaxis

The movement of an organism either towards or away from a source of light. 64

phylogenetic

of or relating to the evolutionary development of organisms. 37

plasticity

the changing of neurons, the organization of their networks, and their function via new experiences. 35, 61

PWM

Pulse Width Modulation. 118, 138

RC

Resistor - Capacitor. 43

RNA

Ribonucleic Acid. 47

RNN

Recurrent Neural Network. 32, 46, 51, 73, 135, 175, 247, 248, 255

SAGA

Species Adaption Genetic Algorithm. 39

sensorimotor

of or pertaining to both sensory and motor activity. 55, 66, 159

somatosensory

of or pertaining to the perception of sensory stimuli produced by the skin or internal organs. 66

Subsumption Architecture

a reactive robot architecture heavily associated with behavior-based robotics. 40

TdGRN

Time-Delayed Gene Regulatory Networks. 50

TDRNN

Time Delay Recurrent Neural Network. 46, 74

TF

Transcription Factor. 48, 49

US

Ultrasound. 159

USB

Universal Serial Bus. 128

WCS

Wisconsin Card Sorting. 66

Bibliography

[1] R.McN. Alexander. The gaits of bipedal and quadrupedal animals. *The International Journal of Robotics Research*, 3(2):49–59, 1984.

[2] K.E. Atkinson. *An Introduction to Numerical Analysis*. John Wiley & Sons, 2nd edition, 1989. ISBN 978-0-471-50023-0.

[3] J. Auerbach and J.C. Bongard. How robot morphology and training order affect the learning of multiple behaviors. In *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, pages 39–46, 2009.

[4] J.M. Baldwin. A new factor in evolution. *The American Naturalist*, 30(354):441–451, 1896.

[5] R.S. Ball. The theory of screws: A study in the dynamics of a rigid body. *Mathematische Annalen*, 9:541–553, 1876.

[6] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In James J. Thomas, editor, *SIGGRAPH*, pages 223–232. ACM, 1989.

[7] J.P. Barreto, A. Trigo, P. Menezes, J. Dias, and A.T. De Almeida. FED-the free body diagram method. kinematic and dynamic modeling of a six leg robot. In *Advanced Motion Control, 1998. AMC '98-Coimbra., 1998 5th International Workshop on*, pages 423 –428, June 1998.

[8] L.W. Barsalou. Perceptual symbol systems. *Behavioral and Brain Sciences*, 22:577–609, 1999.

[9] J. Beal, T. Lu, and R. Weiss. Automatic compilation from high-level biologically-oriented programming language to genetic regulatory networks. *PLoS ONE*, 6(8):e22490, 08 2011.

[10] R.D. Beer. On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(4):469–509, 1995.

[11] R.D. Beer. Parameter space structure of continuous-time recurrent neural networks. *Neural Computation*, 18(12):3009–3051, 2006.

[12] R.D. Beer. *Progress in Motor Control V: A Multidisciplinary Perspective*, chapter Beyond Control: The Dynamics of Brain-Body-Environment Interaction in Motor Systems, pages 7–24. Springer, 2009.

[13] R.D. Beer. Fitness space structure of a neuromechanical system. *Adaptive Behavior*, 2010. (not yet published).

[14] R.D. Beer. *The Horizons for Evolutionary Robotics*, chapter Dynamical analysis of evolved agents: A primer. The MIT Press, 2010. (in press).

[15] R.D. Beer and J.C. Gallagher. Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1(1), 1992.

[16] F. Bellas, J.A. Becerra, J. Santos Reyes, and R.J. Duro. Applying synaptic delays for virtual sensing and actuation in mobile robots. In *IJCNN (6)*, pages 144–152, 2000.

[17] P.J. Bentley. Adaptive fractal gene regulatory networks for robot control. In J. Miller, editor, *Workshop on Regeneration and Learning in*

Developmental Systems, Genetic and Evolutionary Computation Conference (GECCO 2004), volume 154, pages 477+, January 2004.

[18] R.A. Brooks. A robust layered control system for a mobile robot. A. I. Memo 864 864, Massachusetts Institute of Technology Artificial Intelligence Laboratory, September 1985.

[19] R.A. Brooks. Challenges for complete creature architectures. In *In*, pages 434–443. MIT Press, 1990.

[20] R.A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

[21] R.A. Brooks. Integrated systems based on behaviors. *Stanford University*, 2:46–50, 1991.

[22] R.A. Brooks. Intelligence without reason. In Ray Myopoulos, John; Reiter, editor, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 569–595, Sydney, Australia, August 1991. Morgan Kaufmann.

[23] R.A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[24] R.A. Brooks. Artificial life and real robots. In *Proceedings of the First European Conference on Artificial Life*, pages 3–10. MIT Press, 1992.

[25] R.A. Brooks. From earwigs to humans. *Robotics and Autonomous Systems*, 20:291–304, 1996.

[26] J.A. Bullinaria. Lifetime learning as a factor in life history evolution. *Artificial Life*, 15(4):389–409, 2009.

[27] H.J. Cheil and R.D. Beer. *Encyclopedia of Neuroscience*, chapter Computational neuroethology, pages 23–28. Elsevier, 2008.

[28] L. Chen and K. Aihara. Stability of genetic regulatory networks with time delay. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, 49(5):602 –608, may 2002.

[29] D. Cliff, P. Husbands, and I. Harvey. Explorations in evolutionary robotics. *Adaptive Behavior*, 2(1):73–110, June 1993.

[30] J. Clune, B.E. Beckmann, C. Ofria, and R.T. Pennock. Evolving coordinated quadruped gaits with the hyperneat generative encoding. In *Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, CEC'09, pages 2764–2771, Piscataway, NJ, USA, 2009. IEEE Press.

[31] B. Cohen, D. Saad, and E. Marom. Efficient training of recurrent neural network with time delays. *Neural Networks*, 10(1):51 – 59, 1997.

[32] E. Drumwright. A fast and stable penalty method for rigid body simulation. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):231–240, January 2008.

[33] R.J. Duro and J. Santos. ECG beat classification with synaptic delay based artificial neural networks. In José Mira, Roberto Moreno-Díaz, and Joan Cabestany, editors, *Biological and Artificial Computation: From Neuroscience to Technology*, volume 1240 of *Lecture Notes in Computer Science*, pages 962–970. Springer Berlin / Heidelberg, 1997. 10.1007/BFb0032556.

[34] R.J. Duro, J. Santos Reyes, J.A. Becerra, F. Bellas, and J. Luis Crespo. Using higher order synapses and nodes to improve sensing capabilities of mobile robots. In *ESANN*, pages 81–88, 2000.

[35] M.S. Erden and K. Leblebicioglu. Torque distribution in a six-legged robot. *Robotics, IEEE Transactions on*, 23(1):179 –186, feb. 2007.

[36] R. Featherstone. *Rigid Body Dynamics Algorithms*. Kluwer international series in engineering and computer science: Robotics. Springer London, Limited, 2008.

[37] D. Floreano. Evolutionary robotics in behavior engineering and artificial life. In *Evolutionary Robotics: From Intelligent Robots to Artificial Life. Applied AI Systems, 1998. Evolutionary Robotics Symposium*. AAI Books, 1998.

[38] K. Funahashi and Y. Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6(6):801 – 806, 1993.

[39] N.L. Geard and J. Wiles. A gene network model for developing cell lineages. *Artificial Life*, 11(3):249–268, 2005.

[40] R.M. Ghigliazza. *Neuromechanical models for insect locomotion*. PhD thesis, Princeton, 2004.

[41] R.M. Ghigliazza and P. Holmes. Towards a neuromechanical model for insect locomotion: hybrid dynamical systems. *Regular and Chaotic Dynamics*, 10(2):193–225, 2005.

[42] F. Gruau. Automatic definition of modular neural networks. *Adapt. Behav.*, 3:151–183, September 1994.

[43] F. Gruau. *Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm.* PhD thesis, Centre d'étude nucléaire de Grenoble and at l'Ecole Normale Supérieure de Lyon, France, 1994.

[44] K. Gurney. *An Introduction to Neural Networks*. Taylor & Francis, Inc., Bristol, PA, USA, 1997.

[45] I. Harvey. *The Artificial Evolution of Adaptive Behaviour*. PhD thesis, The University of Sussex, April 1995. Originally submitted, September 1993.

[46] I. Harvey. Evolving robot consciousness: The easy problems and the rest. In J.H. Fetzer, editor, *Evolving Consciousness*, pages 205–219. Advances in Consciousness Research Series, John Benjamins, Amsterdam, 2002. ISBN 90 272 5154 1.

[47] I. Harvey, P. Husbands, and D. Cliff. Issues in evolutionary robotics. Cognitive Science Research Paper CSRP 219, The University of Sussex, School of Cognitive and Computing Sciences, 1993.

[48] E. J. Haug. *Computer aided kinematics and dynamics of mechanical systems. Vol. 1: basic methods*. Allyn & Bacon, Inc., Needham Heights, MA, USA, 1989.

[49] G.E. Hinton and S.J. Nowlan. How learning can guide evolution. *Complex Systems*, 1:495–502, 1987.

[50] T. Ho, S. Choi, and S. Lee. Development of a biomimetic quadruped robot. *Journal of Bionic Engineering*, 4(4):193 – 199, 2007.

[51] J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[52] P. Holmes, R.J. Full, D. Koditschek, and J. Guckenheimer. The dynamics of legged locomotion: Models, analyses, and challenges. *SIAM Review*, 48(2):207–304, 2006.

[53] J.J. Hopfield and D.W. Tank. Computing with neural circuits: a model. *Science*, 233(4764):625–633, 1986.

[54] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989.

[55] S. Hu and J. Wang. Global stability of a class of continuous-time recurrent neural networks. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, 49(9):1334–1347, 2002.

[56] J.D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[57] P. Husbands and I. Harvey. Evolution versus design: Controlling autonomous robots. In *Intergrating Perception, Planning and Action: Proceedings of 3rd Annual Conference on Artificial Intelligence, Simulation and Planning*, pages 139–146. IEEE Press, 1992.

[58] P. Husbands, I. Harvey, and D. Cliff. Analysing recurrent dynamical networks evolved for robot control. Cognitive Science Research Paper CSRP 265, School of Cognitive and Computing Sciences, University of Sussex, 1993.

[59] N. Jakobi. Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adapt. Behav.*, 6(2):325–368, 1997.

[60] N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Advances in Artificial Life: Proc. 3rd European Conference on Artificial Life*, pages 704–720. Springer-Verlag, 1995.

[61] M. Joachimczak and B. Wróbel. Evolving gene regulatory networks for real time control of foraging behaviours. In Harold Fellermann, Mark Dörr, Martin M. Hanczyc, Lone L. Laursen, Sarah Maurer, Daniel Merkle, Pierre-Alain Monnard, Kasper Stoy, and Steen Rasmussen,

editors, *Artificial Life XII: Proceedings of the Twelfth International Conference on the Simulation and Synthesis of Living Systems*, pages 348–355, Cambridge, MA, August 2010. MIT Press.

[62] S.A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, USA, 1 edition, June 1993.

[63] B. Kim. Centroid-based analysis of quadruped-robot walking balance. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pages 1 –6, june 2009.

[64] J.F. Knabe, C.L. Nehaniv, M.J. Schilstra, and T. Quick. Evolving biological clocks using genetic regulatory networks. In *In Proceedings of the Artificial Life 10 Conference (Alife X*. MIT Press, 2006.

[65] J.R. Koza. Evolution of subsumption using genetic programming. In *Proceedings of the first European Conference on Artificial Life*, pages 110–119. The MIT Press, 1993.

[66] R. Kram, B. Wong, and R.J. Full. Three-dimensional kinematics and limb kinetic energy of running cockroaches. *J Exp Biol*, 200(13):1919–1929, July 1997.

[67] E.A. Kravitz. Hormonal Control of Behavior: Amines and the Biasing of Behavioral Output in Lobsters. *Science*, 241:1775–1781, September 1988.

[68] R. Kukillaya, J. Proctor, and P. Holmes. Neuromechanical models for insect locomotion: Stability, maneuverability, and proprioceptive feedback. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 19(2):026107, 2009.

[69] S. Kumar. A developmental genetics-inspired approach to robot control. In *Genetic and Evolutionary Computation Conference*, pages 304–309, 2005.

[70] M. Lakshmanan and D.V. Senthilkumar. *Dynamics of Nonlinear Time-Delay Systems*. Springer Series in Synergetics. Springer, 2011.

[71] K.J. Lang, A.H. Waibel, and G.E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Netw.*, 3(1):23–43, January 1990.

[72] X. Li, S. Rao, W. Jiang, C. Li, Y. Xiao, Z. Guo, Q. Zhang, L. Wang, L. Du, J. Li, L. Li, T. Zhang, and Q. Wang. Discovery of time-delayed gene regulatory networks based on temporal gene expression profiling. *BMC Bioinformatics*, 7(1):26+, 2006.

[73] B.S. Lin and S. Song. Dynamic modeling, stability, and energy efficiency of a quadrupedal walking machine. *Journal of Robotic Systems*, 18(11):657–670, 2001.

[74] H. Lipson. *Biomimetics: Biologically Inspired Technologies*, chapter Evolutionary Robotics and Open-Ended Design Automation, pages 129–155. Taylor & Francis, November 2005.

[75] O. Lund, H.H.and Miglino. Evolving and breeding robots. In *Proceedings of the First European Workshop on Evolutionary Robotics*, pages 192–210, London, UK, 1998. Springer-Verlag.

[76] C. Macleod, G. Maxwell, and S. Muthuraman. Incremental growth in modular neural networks. *Engineering Applications of Artificial Intelligence*, 22(4-5):660–666, June 2009.

[77] J.M. Mahaffy and C.V. Pao. Models of genetic control by repression with time delays and spatial effects. *Journal of Mathematical Biology*, 20:39–57, 1984.

[78] M. Malek-Zavarei and M. Jamshidi. *Time-Delay Systems: Analysis, Optimization and Applications*. Elsevier Science Inc., New York, NY, USA, 1987.

[79] M. Maniadakis and J. Tani. Acquiring Rules for Rules: Neuro-Dynamical Systems Account for Meta-Cognition. *Adaptive Behavior*, 17(1):58–80, 2009.

[80] J.L. Meriam and L.G. Kraige. *Engineering Mechanics: Statics*. Wiley and Sons, 2002. ISBN 0-471-40646-5.

[81] B.V. Mirtich. *Impulse-based dynamic simulation of rigid body systems*. PhD thesis, University of California, Berkeley, 1996. AAI9723116.

[82] N.A.M. Monk. Oscillatory expression of $hes1$, p53, and nf-κb driven by transcriptional time delays. *Current Biology*, 13(16):1409 – 1413, 2003.

[83] S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines (Intelligent Robotics and Autonomous Agents)*. The MIT Press, March 2004.

[84] W. Pan, Z. Wang, H. Gao, Y. Li, and M. Du. On multistability of delayed genetic regulatory networks with multivariable regulation functions. *Mathematical Biosciences*, 228(1):100 – 109, 2010.

[85] B.A. Pearlmutter. Dynamic recurrent neural networks. Technical Report v. 90-196, Carnegie Mellon University, Computer Science Department, 1990.

[86] P. Phattanasri, H.J. Chiel, and R.D. Beer. The dynamics of associative learning in evolved model circuits. *Adaptive Behavior*, 15(4):377–396, 2007.

[87] A.S. Potts and J.J. da Cruz. *Mobile Robots - Current Trends*, chapter A Kinematical and Dynamical Analysis of a Quadruped Robot, pages 239–262. InTech, 2011.

[88] S.R. Quartz and T.J. Sejnowski. The neural basis of cognitive development: A constructivist manifesto. *Behavioral and Brain Sciences*, 20(04):537–556, 1997.

[89] J. Reinitz, E. Mjolsness, and D.H. Sharp. Model for cooperative control of positional information in drosophila by bicoid and maternal hunchback. *Journal of Experimental Zoology*, 271(1):47–56, 1995.

[90] A. Riegler. Natural or internal selection? the case of canalization in complex evolutionary systems. *Artif. Life*, 14(3):345–362, 2008.

[91] R.E. Ritzmann, R.D. Quinn, and M.S. Fischer. Convergent evolution and locomotion through complex terrain by insects, vertebrates and robots. *Arthropod Structure & Development*, 33(3):361 – 379, 2004.

[92] D.E. Rumelhart and J.L. McClelland. *Parallel Distributed Processing*, volume 1. MIT Press, 1986.

[93] M.J. Schilstra and H. Bolouri. Modelling the regulation of gene expression in genetic regulatory networks. Technical report, Biocomputation group, University of Hertfordshire., 2002.

[94] C.W. Seys and R.D. Beer. Effect of encoding on the evolvability of an embodied neural network. In *Genetic and Evolutionary Computation Conference (GECCO2006) Workshop Program: Complexity*

through Development and Self-Organizing Representations (CODES-OAR), Seattle, WA, USA, 8-12 July 2006. ACM Press.

[95] C.W. Seys and R.D. Beer. *Advances in Artificial Life*, volume 4648/2007 of *Lecture Notes in Computer Science*, chapter Genotype Reuse More Important than Genotype Size in Evolvability of Embodied Neural Networks, pages 915–924. Springer Berlin / Heidelberg, 2007.

[96] L.P. Shayer and S.A. Campbell. Stability, bifurcation, and multistability in a system of two coupled neurons with multiple time delays. *SIAM Journal on Applied Mathematics*, 61(2):pp. 673–700, 2000.

[97] M. Shein Idelson, E. Ben-Jacob, and Y. Hanein. Innate synchronous oscillations in freely-organized small neuronal circuits. *PLoS ONE*, 5(12):e14443, 12 2010.

[98] A. Silvescu and V. Honavar. Temporal boolean network models of genetic networks and their inference from gene expression time series. *Complex Systems*, 13:2001, 2001.

[99] K. Sims. Evolving 3d morphology and behavior by competition. *Artif. Life*, 1(4):353–372, 1994.

[100] K. Sims. Evolving virtual creatures. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22, New York, NY, USA, 1994. ACM.

[101] P. Smolen, D.A. Baxter, and J.H. Byrne. Modeling transcriptional control in gene networks - methods, recent results, and future directions. *Bulletin of Mathematical Biology*, 62:247–292, 2000.

[102] R. Somogyi and C. Sniegoski. Modeling the complexity of genetic networks: Understanding multigenic and pleiotropic regulation. *Complexity*, 1:45–63, 1996.

[103] S. Soyguder and H. Alli. Kinematic and dynamic analysis of a hexapod walking-running-bounding gaits robot and control actions. *Computers & Electrical Engineering*, 38(2):444 – 458, 2011.

[104] K.O. Stanley. *Efficient Evolution of Neural Networks Through Complexification*. PhD thesis, Department of Computer Sciences, The University of Texas at Austin, 2004.

[105] K.O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines*, 8(2):131–162, June 2007.

[106] K.O. Stanley, D.B. D'Ambrosio, and J. Gauci. A hypercube-based indirect encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212, April 2009.

[107] K.O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[108] K.O. Stanley and R. Miikkulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130, 2003.

[109] G.F. Striedter. *Principles of Brain Evolution*. Sinauer Associates, 1 edition, October 2004.

[110] S.H. Strogatz. *Nonlinear Dynamics And Chaos: With Applications To Physics, Biology, Chemistry, And Engineering (Studies in nonlinearity)*. Studies in nonlinearity. Perseus Books Group, 1 edition, January 1994.

[111] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[112] M. Taylor, S. Whiteson, and P. Stone. Comparing evolutionary and temporal difference methods for reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1321–28, July 2006.

[113] M.A. Trefzer, T. Kuyucu, J.F. Miller, and A.M. Tyrrell. Evolution and analysis of a robot controller based on a gene regulatory network. In *Proceedings of the 9th international conference on Evolvable systems: from biology to hardware*, ICES'10, pages 61–72, Berlin, Heidelberg, 2010. Springer-Verlag.

[114] E. Tuci, M. Quinn, and I. Harvey. An Evolutionary Ecological Approach to the Study of Learning Behavior Using a Robot-Based Model. *Adaptive Behavior*, 10(3-4):201–221, 2002.

[115] F.J. Varela, E.T. Thompson, and E. Rosch. *The Embodied Mind*. The MIT Press, 1991.

[116] C. Vella. *Gravitas: An extensible physics engine framework using object-oriented and design pattern-driven software architecture principles*. Masters in information technology, University of Malta, Msida, 2008.

[117] J. Vohradsky. Neural network model of gene expression. *The FASEB Journal*, 15(3):846–854, 2001.

[118] Y. Wang, Z. Wang, and J. Liang. On robust stability of stochastic genetic regulatory networks with time delays: A delay fractioning approach. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 40(3):729 –740, june 2010.

[119] Z. Wang, H. Gao, J. Cao, and X. Liu. On delayed genetic regulatory networks with polytopic uncertainties: Robust stability analysis. *NanoBioscience, IEEE Transactions on*, 7(2):154 –163, june 2008.

[120] R. Wehner. Matched filters: neural models of the external world. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, Volume 161(4):511–531, July 1987.

[121] R. Wehner and R. Menzel. Do insects have cognitive maps? *Annual Review of Neuroscience*, 13:403–414, 1990.

[122] A. Wuensche. Genomic regulation modeled as a network with basins of attraction. In *Proc. Pac. Symp. Biocomput.*, pages 89–102. World Scientic Publishing, 1998.

[123] B. Yamauchi and R.D. Beer. Integrating reactive, sequential, and learning behavior using dynamical neural networks. In *SAB94: Proceedings of the third international conference on Simulation of adaptive behavior : from animals to animats 3*, pages 382–391, Cambridge, MA, USA, 1994. MIT Press.

[124] C. Yeang and T. Jaakkola. Time series analysis of gene expression and location data. In *Bioinformatics and Bioengineering, 2003. Proceedings. Third IEEE Symposium on*, pages 305 – 312, march 2003.

[125] J. Yosinski, J. Clune, D. Hidalgo, S. Nguyen, J.C. Zagal, and H. Lipson. Evolving robot gaits in hardware: the hyperneat generative encoding vs. parameter optimization. In *Proceedings of the 20th European Conference on Artificial Life*, August 2011.

[126] C.H. Yuh, H. Bolouri, and E.H. Davidson. Genomic cis-regulatory logic: Experimental and computational analysis of a sea urchin gene. *Science*, 279(5358):1896–1902, 1998.

[127] C.H. Yuh, H. Bolouri, and E.H. Davidson. Cis-regulatory logic in the endo16 gene: switching from a specification to a differentiation mode of control. *Development*, 128:617–629, 2001.

[128] P. Zahadat, T. Schmickl, and K. Crailsheim. Evolving reactive controller for a modular robot: Benefits of the property of state-switching in fractal gene regulatory networks. In Tom Ziemke, Christian Balkenius, and John Hallam, editors, *From Animals to Animats 12*, volume 7426 of *Lecture Notes in Computer Science*, pages 209–218. Springer Berlin Heidelberg, 2012.

[129] F.Y. Zhang and H.F. Huo. Global stability of hopfield neural networks under dynamical thresholds with distributed delays. *Discrete Dynamics in Nature and Society*, 2006:Article ID 27941, 11 pages, 2006.

Vita

Benjamin John Derrick was born in Southampton, Hampshire on March the 2nd, 1986 to Simon and Fiona Derrick. He went to King Edward VI School in Southampton and partook in the Royal Academy of Engineering Year in Industry scheme with a placement at Flight Refuelling Ltd in Wimborne, part of Cobham PLC. He graduated with a First Class Honours degree in Mechanical Engineering in 2009 from the University of Durham and was awarded a studentship for postgraduate work.

This Thesis was typeset with LaTeX2e by the author. Graphics were generated with Inkscape, or the Python module `matplotlib` [56].